# NAVAL POSTGRADUATE SCHOOL
# Monterey, California

# THESIS

**REENGINEERING THE UNITED STATES MARINE CORPS' RECRUIT DISTRIBUTION MODEL (RDM)**

by

Kevin J. Snoap

September 1998

| | |
|---|---|
| Thesis Advisor: | Hemant K. Bhargava |
| Associate Advisor: | Suresh Sridhar |

**Approved for public release; distribution is unlimited.**

# REPORT DOCUMENTATION PAGE

*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

| | | |
|---|---|---|
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE**<br>September 1998 | **3. REPORT TYPE AND DATES COVERED**<br>Master's Thesis |

| | |
|---|---|
| **4. TITLE AND SUBTITLE : REENGINEERING THE UNITED STATES MARINE CORPS' RECRUIT DISTRIBUTION MODEL (RDM)** | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)**<br>Kevin J. Snoap | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES**

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

| | |
|---|---|
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT**<br>Approved for public release; distribution is unlimited. | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT** ***(maximum 200 words)***

The United States Marine Corps accomplishes its mission "to put the right Marine in the right place at the right time with the right skills and quality of life" in a variety of ways. One of the information systems assisting the Marine Enlisted Assignments branch is the Recruit Distribution Model (RDM). This thesis proposes changes to the RDM user interface, data management, assignment model, and analysis capability. With the use of business process reengineering, process modeling, mathematical modeling, and database design a fully functional prototype has been developed to address each identified change proposal. This reengineered system includes numerous innovations such as an intuitive navigational scheme using switchboards, and the elimination of manual data entry for data already available in the system. It also provides a number of significant contributions beneficial to the USMC. For instance, the reengineered system allows the user to objectively analyze different results by comparing four different objective measures, and its mathematical model uses commercial-off-the-shelf products eliminating a proprietary solver. All these changes will empower managers to effectively and efficiently manage the assignment of recruits in order to meet the challenges of the 21st century.

| | |
|---|---|
| **14. SUBJECT TERMS**<br>USMC, Databases, Manpower Assignment, Models, Decision Support Systems, Graphical User Interface | **15. NUMBER OF PAGES** 100 |
| | **16. PRICE CODE** |

| | | | |
|---|---|---|---|
| **17. SECURITY CLASSIFICATION OF REPORT**<br>Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE**<br>Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT**<br>Unclassified | **20. LIMITATION OF ABSTRACT**<br>UL |

# REENGINEERING THE UNITED STATES MARINE CORPS' RECRUIT DISTRIBUTION MODEL (RDM)

Kevin J. Snoap
Lieutenant, United States Navy
B.S., University of Texas at Austin, 1991

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 1998**

Author: Kevin J. Snoap
Kevin J. Snoap

Approved by: HKBhargava
Hemant K. Bhargava, Advisor

Suresh Sridhar, Associate Advisor

Rueben T. Harris, Chairman, Department of Systems Management

# ABSTRACT

The United States Marine Corps accomplishes its mission "to put the right Marine in the right place at the right time with the right skills and quality of life" in a variety of ways. One of the information systems assisting the Marine Enlisted Assignments branch is the Recruit Distribution Model (RDM). This thesis proposes changes to the RDM user interface, data management, assignment model, and analysis capability. With the use of business process reengineering, process modeling, mathematical modeling, and database design a fully functional prototype has been developed to address each identified change proposal. This reengineered system includes numerous innovations such as an intuitive navigational scheme using switchboards, and the elimination of manual data entry for data already available in the system. It also provides a number of significant contributions beneficial to the USMC. For instance, the reengineered system allows the user to objectively analyze different results by comparing four different objective measures, and its mathematical model uses commercial-off-the-shelf products eliminating a proprietary solver. All these changes will empower managers to effectively and efficiently manage the assignment of recruits in order to meet the challenges of the 21st century.

# TABLE OF CONTENTS

# LIST OF FIGURES

# I. INTRODUCTION

## A. THESIS PURPOSE

The main purpose of this thesis was to reengineer a United States Marine Corps' Manpower Assignment model concerned with the distribution of recruits to schools. This model is called the Recruit Distribution Model (RDM) [Ref. 1]. Throughout this thesis, the RDM is addressed as either RDM or the old system.

Additionally, an important purpose of this thesis was to build a functional prototype of the reengineered RDM. The new system is called the Recruit Distribution Decision Support System (RDdss). It demonstrates the functionality of the reengineered RDM. Throughout this thesis, the RDdss is addressed as either RDdss or the new system.

The majority of this thesis is devoted to a discussion of the RDdss. As necessary, the RDM is discussed. The following section is a general discussion designed to set the stage for understanding the rest of the thesis. It provides a problem description of recruit distribution in the United States Marine Corps (USMC). The last section of this chapter discusses the significant contributions of this thesis.

## B. GENERAL PROBLEM DESCRIPTION

Recruit distribution in the USMC is the process that assigns recruits to an entry level school (ELS) leading to a military occupational specialty (See Figure 1). These assignments are made about 48 times a year, during the last week of Marine Corps Recruit Depot (MCRD) training. In this ending period of the MCRD, the recruits are facing the "Crucible," which is the final wicket a recruit endures before officially becoming a Marine. Consequently, the use of the titles recruit and Marine are used interchangably in this paper.

For at least two reasons, this assignment process is a critical manpower function. First, a Marine's military occupational specialty (MOS) ultimately determines

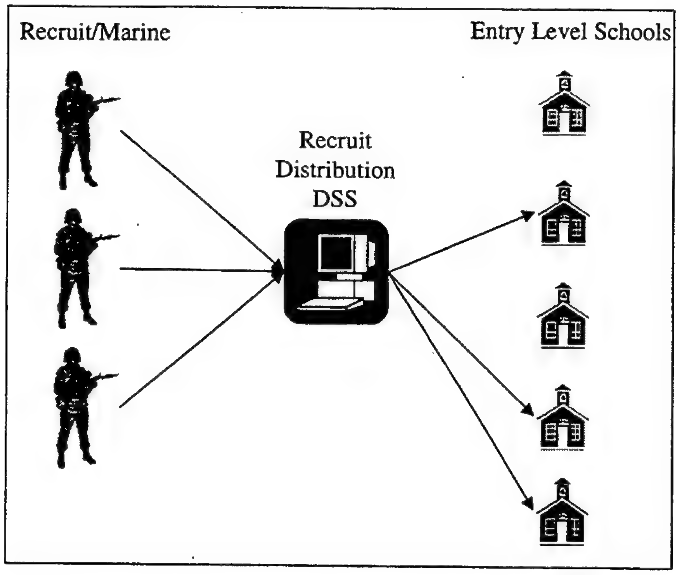


Figure 1. Assigning Marines to Schools

the member's career. Therefore, it is in the best interest of both the Marine and the USMC that a school assignment matching the Marine's desire is made. Second, the success obtained by a service member during his or her time in the USMC is partially based upon successful completion of their ELS, where a pattern of success is established. Therefore, a school assignment maximizing the chances of the Marine completing their training is important. So, fulfilling the Marine's desire and matching him or her to an MOS (i.e. ELS) that fits their personal characteristics is critical to the overall health of the USMC, making the assignment process a critical manpower function [Ref. 2].

Fulfilling the desire of the Marine is accomplished through the use of a contract guarantee. This is called a program enlisted for (PEF), and is specified during the recruiting process. For instance, a PEF = 19 is the "Tank and Assault Amphibian Option." There are currently two ELS's associated with this, "M1A1 Tank Crewman" and "Assault Amphibian Crewman." So, a Marine who chose the PEF = 19 is a possible candidate for these two schools and no others.

Once the Marine's school options are known, a Marine-to-school fit is determined for each of these schools (See Chapter III, Section B for details). This fitness determination is partly made by looking at each ELS's minimum eligibility requirements, which are call mandatory properties. This term "property" is used as defined by Webster's Ninth New Collegiate Dictionary: "a quality or trait belonging and especially peculiar to an individual or thing." Examples are Age > 18, Clerical >= 80, and Electrical >= 95. The meaning of the first example is obvious. The other two are based on test scores from the Armed Services Vocational Aptitude Battery (ASVAB) test.

In addition to the mandatory properties, most schools also specify desirable properties. A desired property is the same as a mandatory property, except they are not prerequisites for attending the school. For example, the Traffic Management Coordination school desires Marines with a Clerical score of at least 100. So, a desired property of Clerical >= 100 is specified for this ELS.

By using the information obtained from the PEF, mandatory and desired properties, a fitness matrix is generated. This shows the fit of every Marine to every school he or she is eligible for. Since there are about 100 schools and on average 700 Marines considered for every run, this matrix has the potential of 70,000 matches.

However, the matrix size is actually bigger. This is because each of the schools is broken down by classes. Some ELS's have a class starting each week, others every month, and others every quarter. Over a given year, these classes total about 1,800. Following the practice of the USMC, only the classes over the next 3-4 months are considered during the assignment process (see Figure 2). Therefore, the fitness matrix is increased to a potential size of 350,000.

The fact that the school classes start at different times throughout the year and that the assignment process is only conducted 48 times a year (normally this occurs on friday) causes the "3-month look ahead" in the model and has implications important to the USMC. The concern is school seats which never get filled. Each seat is prepaid, guaranteeing its availability to the USMC. This means every vacant

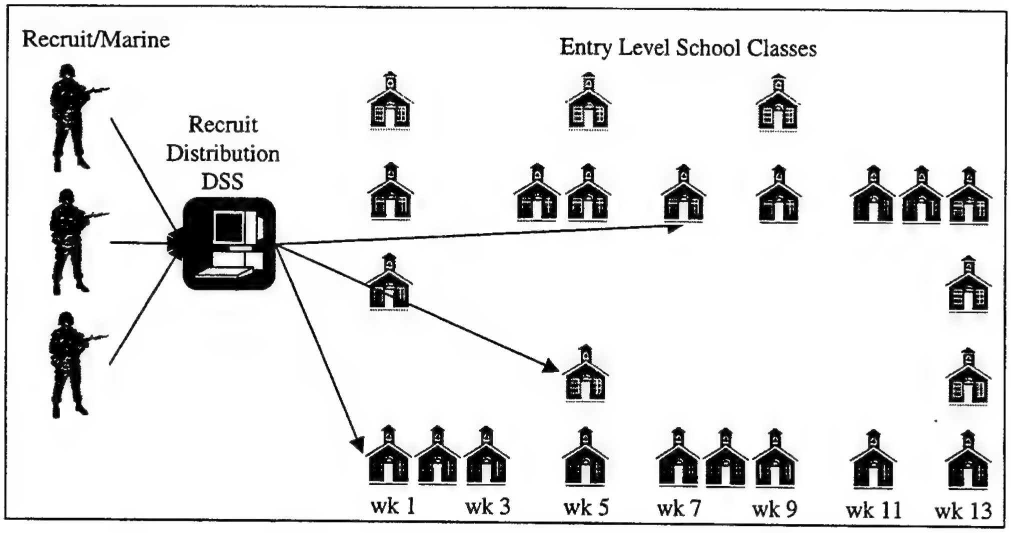


Figure 2. Assigning Marines to School Classes

spot is potentially a lost resource to the USMC. Therefore, in addition to the Marine-to-school fitness concern, there is also the concern of filling all available school seats before their report date is passed.

Another concern is the problem of unassigned Marines. There are numerous reasons why they may not get assigned. Maybe the only school class the Marine was eligible for is already full. Or, possibly he or she is not qualified for any of the school's promised by their PEF. A third possibility is errors in the data provided to the assignment process. Regardless of the cause, it requires identification and corrective action.

Finally, a discussion of the internal and external stakeholders associated with the USMC recruit distribution is given. Internally, there are the USMC and the recruits. Since their concerns were discussed above, they are not given any further consideration. Externally, there is the contractor Decision Support Associates, Inc. (DSAI) who has maintained and upgraded the RDM for over 30 years. They also maintain about eight other major systems for the USMC. For their services, the USMC pays a significant amount of money each year. Additionally, DSAI has proprietary software in some of these systems, which has locked-in the USMC to this company. Besides the contractor, there is one last significant stakeholder, the American taxpayer. In a time of increasing fiscal constraints and shrinking budgets, it is imperative that wise decisions are made in regards to assigning Marines to ELS's. The American taxpayers expect nothing less.

## C. SIGNIFICANT CONTRIBUTIONS

Four significant contributions beneficial to the USMC have been made in this thesis. A detailed discussion of these is given throughout the next three chapters. Here, we list and summarize the contibutions.

- Analysis and articulation of the recruit distribution process - by interviewing USMC and DSAI personnel, reviewing available documentation, and operating the RDM we were able to articulate an understanding of the recruit distribution process using IDEF process modeling. Additional data modeling was articulated in a third normal form relational schema [Ref. 3].
- Development of a mathematical model - by analyzing the assignment process, criteria, and constraints, USMC policies and objectives, a mathematical model for the new system was developed.
- Fully functional prototype - an intuitive and easy to understand new system that seamlessly interfaces with the old system was built. It provides an objective means of comparing assignment results of both systems, and was developed using commercial-off-the-shelf (COTS) software applications.
- Elimination of the proprietary solver and associated contractor lock-in - this was accomplished by replacing the proprietary solver with two COTS applications.

# II. REENGINEERING MOTIVATIONS AND PROCESS VIEW

The RDM has a number of limitations which motivated the USMC to reengineer it. We start this chapter by listing and describing each identified limitation. Then, to develop a better understanding of recruit distribution in the USMC, two different process views are examined. The first is concerned with the RDM operating environment, and the second is concerned with the first level of the RDdss IDEF0 model.

## A. LIMITATIONS OF OLD SYSTEM

Throughout this reengineering effort, a number of limitations of the old system have become apparent. Many of these limits were recognized earlier, and were part of the USMC's motivation to reengineer the RDM. The following is a list of the identified limitations of the old system.

- Navigation
- Data management
- Assignment procedure
- Transaction processing approach

A description of each limitation is now given.

### 1. Navigation

Navigating through the RDM is not intuitive. It is neither obvious where one should start or where one should go. Navigation is accomplished by initially selecting an option from one of the main display's drop down menus. This normally results in a window appearing on the computer's desktop. Then, by pointing and clicking on the displayed window's buttons, further navigation is accomplished. After working with the RDM for a number of hours, we were able to navigate through the application to find and display specific windows. However, after a week or two of not using the system, we had difficulty finding our way around again. In the RDdss, we have created an intuitive navigational scheme using switchboards.

### 2. Data Management

Data management in the RDM is poor, leading to the introduction of numerous errors. The biggest problem in this regard deals with data entry. The RDM violates the basic rule of never requiring the user to enter data already in the system [Ref. 4]. For example, creating a new school in the RDM requires the user to type data into seven different data fields. Only the course identification number has been automated by a drop down list. In the RDdss, when creating a new school we have reduced the manual data entry to one field.

### 3. Assignment Procedure

There are at least five limitations associated with the old system's assignment procedure. It

- is encoded into proprietary software,
- examines schools sequentially, rather than globally,
- makes assignments based on school priority rather than weights,
- attempts to maximize fill rather than fit-and-fill, and
- is relatively inflexible.

The solver performing these assignments was designed back in the 1950's, where the major concern was speed and using the minimal amount of memory. It is written in Fortran, and is proprietary code owned by DSAI. It does not search repetitively for an optimal solution by trying to maximize or minimize an objective function. Instead, it maximizes the fill of prioritized school seats. In the RDdss, we use a well know algorithm called CPlex [Ref. 5]. After conducting around 600 iterations of comparing 344 Marines to 576 school classes, it produces an *optimal* solution. Its objective function was written to make the assignment procedure flexible. It allows the RDdss manager to "game" the system by making fit-and-fill trade-offs, until a "good" solution is found.

### 4. Transaction Processing Approach

Finally, the old system follows a transaction processing approach, vice a decision support approach [Ref. 6]. This is partially due to the inflexibility of the solver. However, another contributor is the extreme difficulty in comparing one run to another. Other than providing a means for manually computing the numerical difference in Marines assigned, the RDM provides little insight or information views for comparing runs. This is because there is no way to objectively compare one run to another. As it is now, if the RDM manager launches the assignment model and everybody is assigned, the result becomes the approved assignments. In the RDdss, we have created an entire process devoted to providing insightful analysis of a given run (See Chapter IV, Section D for details). Its purpose is to support the making of wise assignment decisions.

## B. A PROCESS VIEW

To develop a better understanding of recruit distribution in the USMC, two different process views are now examined. The first is concerned with the current operating environment of the old system. The goal is to develop a big picture view of this process. For those interested in further study of the old system, Appendix A contains the RDM business process IDEF0 model.

The second process view examined concerns the first level of the RDdss business process IDEF0 model. The goal is to develop a better understanding of the new system, without going into great detail. This lays a good foundation for the RDdss discussions in the remainder of this thesis. Appendix B contains the entire IDEF0 model of the new system.

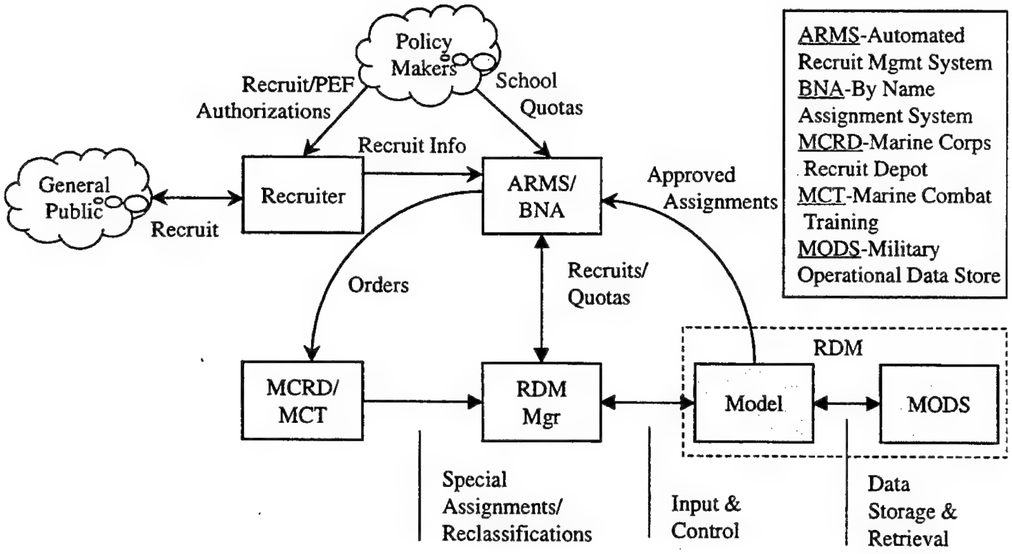

Policy Makers
Recruit/PEF Authorizations
School Quotas
Recruit Info
General Public
Recruiter
Recruit
ARMS/ BNA
Approved Assignments
Orders
Recruits/ Quotas
MCRD/ MCT
RDM Mgr
RDM
Model
MODS
Special Assignments/ Reclassifications
Input & Control
Data Storage & Retrieval
ARMS-Automated Recruit Mgmt System
BNA-By Name Assignment System
MCRD-Marine Corps Recruit Depot
MCT-Marine Combat Training
MODS-Military Operational Data Store

Figure 3. Current Recruit Distribution Operating Environment

### 1. Recruit Distribution Model Operating Environment

Figure 3 graphically depicts the operating environment of the USMC recruit distribution. At the top are the policy makers, who forecast and decide how many recruits and corresponding school seats are needed over the next few years. In addition to this, they determine how many PEF's or school guarantees to make available for a given year. Currently, about 65-70% of the recruits enter the USMC with a quarantee. The recruit and PEF authorizations for the following year are given to the USMC recruiters, and the school seats or quotas determined for the following years is input into the By Name Assignment (BNA) system.

The recruiters use the recruit and PEF authorizations in recruiting from the general public. Once a potential candidate is found they take the ASVAB, if they have not already done so. Based on the results, the recruiter can offer different quarantees. Once the candidate signs a contract with the USMC, their test scores, PEF, and other personal information such as age and height are entered into the Automated Recruit Management System (ARMS).

Both the ARMS and BNA systems utilize large main frame databases. They serve as central repositories for maintaining data on many aspects dealing with the USMC. Keeping this information stored in one location ensures everyone is using the same data, providing consistency for all users. One user is the RDM manager, who retrieves recruit and school information from these systems for use in the RDM.

Additionally, the RDM receives data from two other sources. The MCRD instructors provide special assignment inputs. These are personnel identified as having the talents or abilities well suited for a particular school. The other data comes from the MCT. They provide the RDM manager with reclassification information. For instance, a Marine is reclassified if he or she is injured during MCT and is unable to make the start date of their assigned ELS.

All this recruit and school data is input into the RDM, where it is stored in the Military Operational Data Store (MODS). The model is then run. Once a satisfactory set of assignments is obtained, the RDM manager uploads the approved assignments to the ARMS and BNA systems. From this assignment information, the MCRD generates orders for the Marines graduating from the MCRD.

### 2. New System Level 0 Diagram

Focus is now shifted to the examination of the first process level of the RDdss. Looking any deeper will provide more detail than is necessary at this point. As mentioned earlier, the intent is to lay the foundation for the RDdss discussions in the remainder of this thesis.

Figure 4 shows the level 0, or context diagram of the RDdss. The program used to generate this illustration was BPWin, which is based on IDEF0 modeling. Consequently, all arrows entering from the left are considered"inputs," arrows entering from the top are "controls," arrows exiting to the right are "outputs," and arrows entering from the bottom are "mechanisms." A convenient acronym for this is ICOM [Ref. 7].

- I = Input: something consumed in the process
- C = Control: a constraint on the operation of the process

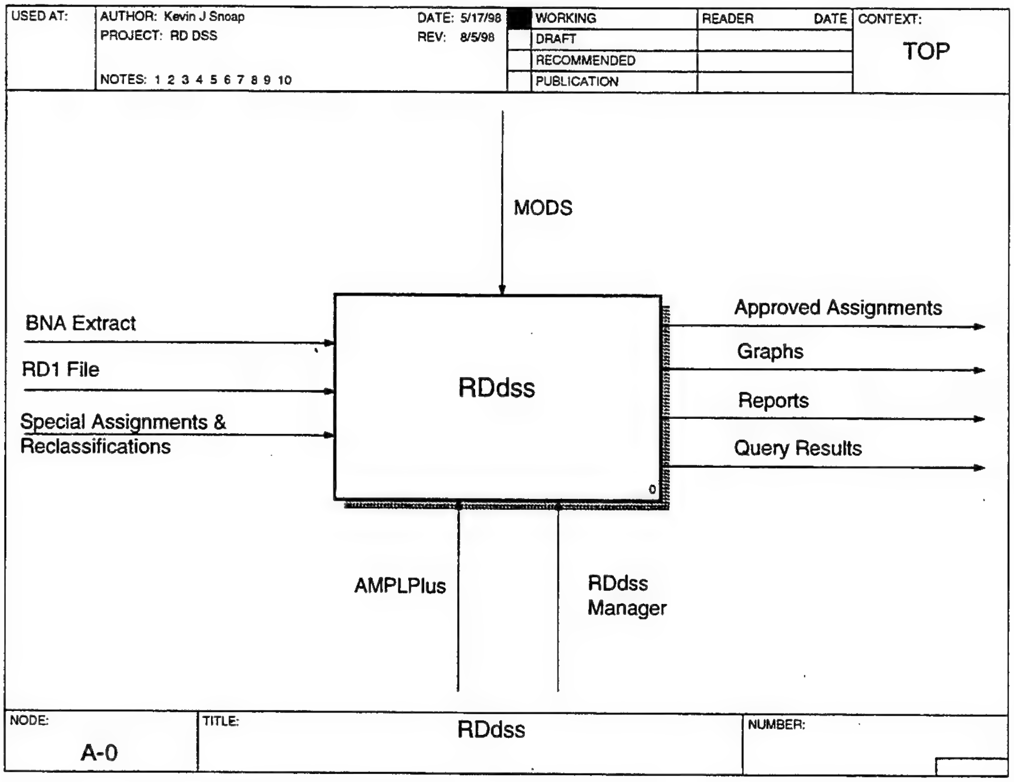


Figure 4. RDdss Process, Level 0 Diagram

- O = Output: something resulting from the process
- M = Mechanism: something used to perform the process, but is not itself consumed

There are three inputs to the RDdss, the BNA extract, the RD1 file, and the special assignments/reclassifications. The first of these contains a set of school class quotas covering a 120 day period. This comes to the RDdss manager as a fixed-width delimited text file. The RD1 file contains the data for the recruits soon to graduate from the MCRD. It also is a fixed-width delimited text file. The special assignments and reclassifications are the recruits identified with special abilities or talents and the Marines requiring reassignment to a different school class, respectively. This last data comes to the RDdss manager as a memo and not a text file.

The only control is the database (or Military Operational Data Store) used to store the data for the RDdss. Among other things, the MODS contains the information specifying the eligibility requirements for each of the schools. This is why it is considered a control. It contains the data controlling or constraining the assignment of Marines to schools.

Besides the approved assignments that were mentioned earlier, the output consists of graphs, reports, and query results. Each graph was designed to provide insight into the current assignment result, plus provide a means for comparing different runs. The reports provide a print-out of information the RDdss manager might find useful. For instance, an approved assignment report and an unassigned Marine report are both available. The query results provide different views of the data in the MODS. For instance, one of the queries provides a listing of the fitness scores of all schools a Marine is eligible for.

Finally, the two mechanisms of the RDdss are the RDdss manager and the commercial-off-the-shelf (COTS) application AMPLPlus. Both of these entites are necessary to make the system work, and are not consumed by the process itself. This is why they are considered mechanisms. The RDdss manager is the one who points and clicks on the buttons of the RDdss, making it operate. AMPLPlus is the application containing the mathematical model used in making optimal Marine-to-school assignments. It interfaces with the CPlex algorithm that actually performs the optimization.

# III. MODELING THE RECRUIT DISTRIBUTION PROBLEM

We model the recruit distribution problem as an assignment problem. Various submodels are developed to compute input paramters and perform other preprocessing steps for the assignment model, which optimally assigns recruits to a school class. Following is a discussion of important points concerning the recruit distribution problem.

1. Schools and Classes: there are about 130 schools per fiscal year, which are broken down by classes (about 1800 classes per fiscal year). Classes of the same school, commencing on different dates, are identical. However, a class commencing soon after a recruit is available for training has greater utility than another class starting at a later date. On average, there are 600 classes in a run.
2. Program Enlisted For (PEF)
   (a) 65-70% of enlistees enter the USMC under a guarantee (or PEF). The remaining recruits enter under an open contract (PEF=00).
   (b) A recruit can be assigned only to schools associated with his or her PEF. Some PEF's are associated with one school, while others are associated with many schools.

## A. OPTIMIZATION OBJECTIVES: FIT AND FILL

Our approach in the assignment model is to optimize by looking at both fit and fill. The importance of both these objectives was discussed in Chapter II, Section B. The idea is to allow the RDdss manager to make a tradeoff between fit and fill.

The high-level objective function is:

- Maximize

$$K_1 \cdot Fitness - K_2 \cdot Penalty$$

Where $K_1$ is the fitness coefficient and $K_2$ is the fill coefficient.

By assigning different values to these coefficients, the RDdss manager is able to make trade-offs between fit and fill. This capability makes the assignment model flexible. It provides the RDdss manager a means for "gaming" the system.

## B. MEASURING FITNESS

The first half of the high-level objective function deals with maximizing the fitness of Marines to schools. Therefore, we cover this topic now. Following is a discussion of important points concerning the measuring of fitness.

1. Marines are only eligible for assignment to schools corresponding to their PEF guarantees.

2. Associated with each school are specific mandatory properties, or minimum eligibility requirments (e.g. AGE $>$ 18).

3. Eligibilty of a Marine to a school is pre-determined by comparing a Marine's attributes (e.g. Age = 20) to the school's mandatory properties. A Marine is eligible only if he or she meets all the minimum school requirements.

4. Many schools also have desirable properties associated with them. This is a property which is desired in a Marine (e.g. Height $>$ 65 inches), but are not prerequisties for attending the school. Since some desirable properties are more desirable than others, a means of distinquishing the properties is necessary. We have followed the Marine Corps practice of using 6 levels of desirability. Level 1 properties are the most desired, followed by level 2, etc.

5. Let $\mathcal{P}(s)$ denote the set of desirable properties for school $s$, and let $level(p)$ denote the desirability level of property $p$. Let $possesses_{(m,p)} = 1$ if Marine $m$ possesses property $p$ (0 otherwise). Further, let $Score_{level(p)}$ ($> 0$) be the score assigned for possessing a $level(p)$ property, where $Score_{level(p)}$ is calculated using an exponential function inversely proportional to $level(p)$. The scores for levels 1 through 6 are shown in Figure 5.

6. The total fitness score ($fit_{m,s}$) of Marine $m$ to school $s$ is composed of two parts: the score ($ManScore_{m,s}$) for possessing mandatory properties, and the score ($DesScore_{m,s}$) for possessing some or all of the desirable properties.

   Our procedure for computing fitness scores is designed in a manner that, for each school, the average fitness score—with the average computed over all qualified Marines, i.e. Marines who have met the mandatory properties—is constant (i.e. 100). The reason we did this was to ensure each school

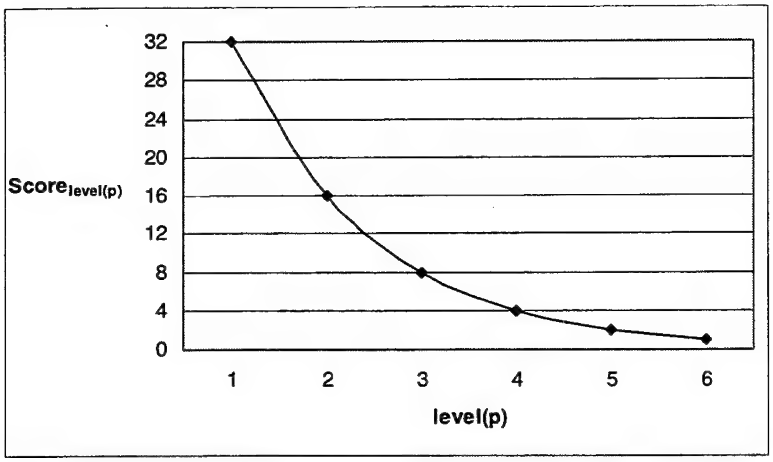


Figure 5. Exponential Function for Calculating $Score_{level(p)}$

received equal treatment in the model, regardless of the number of desirable properties specified for each school. Otherwise, all other things being equal, the school with the greatest number of desirable properties would receive the most assignments.

7. For each school $s$, percentage weights $ManWt_s$ and $DesWt_s$ are assigned, respectively, for the mandatory and desirable properties. For example, for school $s_1$, a weight of 70% may be assigned to the mandatory properties and 30% for the desirable properties.

8. A Marine possessing all mandatory properties for school $s$ is given an initial score $ManScore_{m,s} = ManWt_s \cdot 100$. Marines who do not possess all mandatory properties for a school are given an overall fitness score of 0 for that school.

9. Of all the Marines possessing the mandatory properties, those possessing desirable properties are awarded additional points, which are weighted by the level of the property. The Marine's score for desirable properties is computed as follows:

   - The absolute score $Abs_{m,s}$, for Marine $m$ and school $s$, based on desirable properties is

   $$Abs_{m,s} = \sum_{p \in \mathcal{P}(s)} Score_{level(p)} \cdot possesses(m,p)$$

   The average of these absolute scores is computed over all qualified Marines. Let $AveAbs_s$ denote the average absolute score for school $s$.

- Each Marine's fitness score for desirable properties is then computed as a fraction of this average, and normalized by multiplying with the percentage weight for desirable properties. That is, $DesScore_{m,s}$ is a weighted relative score:

$$DesScore_{m,s} = (\frac{Abs_{m,s}}{AveAbs_s}) \cdot DesWt_s \cdot 100$$

  The overall average of all these scores is, due to the above construction, $DesWt_s \cdot 100$.

10. The final fitness score, $fit_{m,s}$ is simply the sum $ManScore_{m,s} + DesScore_{m,s}$. It may be seen that, for each school, this number averages (over all the qualified Marines for that school) to 100.

11. As mentioned earlier, schools are broken down by classes that are identical, except for their start dates. Therefore, let $fit_{m,s} = fit_{m,c}$ for each class $c$ of school $s$.

## C. MEASURING FILL

The second half of the objective function deals with maximizing the fill of school seats. We now turn our attention to this topic.

As we discussed in Chapter II, school seats are paid for in advance. This is to guarantee the seat is available to the USMC. Consequently, every vacant spot is potentially a lost resource. Since the model is run about once every week, the biggest concern is the unfilled seats having report dates within the next seven days. Conversely, a seat having a report date in three months is not so critical.

To capture the essence of filling school seats with early report dates, we use a penalty function. The idea is simple. School seats having an early report date get a high penalty, and those having a late report date get a low penalty. A number of functions would have worked for this. We have graphed two candidates in Figure 6. The first is a linear function and the second a large-step function. Each of these was tried in the prototype. In our opinion the latter one is the best choice. It gives a lot of emphasis to the first week, plus it treats all schools having report dates within the same week equally. It is the function currently implemented in the rddss.

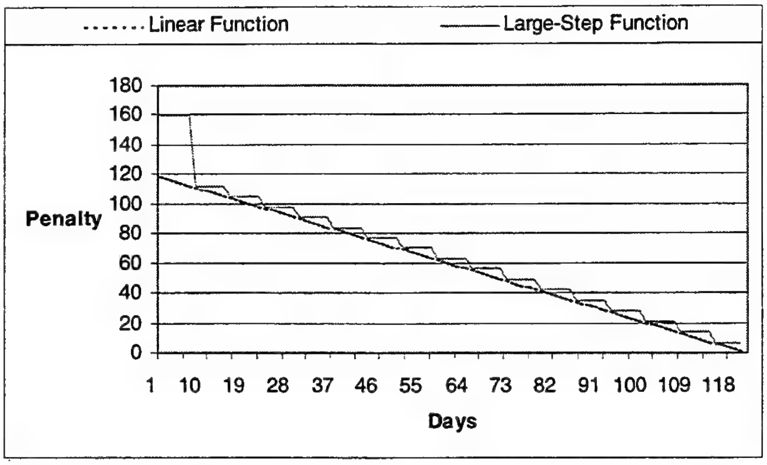


Figure 6. Two Candidate Penalty Functions

## D. ASSIGNMENT MODEL

We are now ready to formally describe the assignment model. We will first specify the model's assumptions. Then, we will list the model's notation, followed by its objective function and constraints.

### 1. Assumptions

1. Each school is unique and identifies a single course of instruction leading to an MOS.

2. In any given run, a school may offer several classes that start on different dates. It is more desirable to fill seats in classes starting earlier. Such classes have a higher penalty per unit shortfall than similar classes starting at a later date.

3. All Marines available for assignment are graduating from the MCRD on the same date.

4. A Marine's eligibility for a school, as well as the fit for each school, is determined by the model preprocessor, taking into account the PEF guarantees, and the mandatory and desirable properties.

5. A Marine is assigned to a school corresponding to their PEF code, or not at all.

6. It is better to leave a Marine unassigned than to overfill a school.

7. The demand for the number of Marines to be trained in each school class is determined by looking at the BNA extract. This demand statement is already constrained by the capacity constraint at each school.

## 2. Notation

- Sets

  $\mathcal{M}$ : Marines

  $\mathcal{C}$ : Classes

- Exogenous Variables (Parameters in AMPL)

  $fit_{m,c}$: the desirability of assigning Marine $m$ to class $c$ (note: the fitness score is zero for Marine-class pairs where either the Marine does not meet the classes mandatory properties, or where the class does not fulfill the Marine's PEF guarantee.)

  $demand_c$: demand for Marines to be trained at class $c$

  $penalty_c$: penalty for each unit of demand not met (higher value means it is more critical to fill the class)

- Decision Variables

  $x_{m,c}$ (binary integer): 1, if Marine is assigned to the given class; 0 otherwise

## 3. Objective

Maximize the Total Utility: (Total Reward - Total Penalty)

$$TU = k_1 \left( \sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}} fit_{m,c} \cdot x_{m,c} \right) - k_2 \left( \sum_{c \in \mathcal{C}} penalty_c \cdot (demand_c - \sum_{m \in \mathcal{M}} x_{m,c}) \right) \quad \text{(III.1)}$$

## 4. Constraints

- assignmentLimit: a Marine is assignable to at most one school

$$\sum_{c \in \mathcal{C}} x_{m,c} \leq 1 \qquad \forall m \qquad \text{(III.2)}$$

- eligibility: a Marine is only assignable to a class they are fit for (this prevents assigning Marines with a fitness score of zero)

$$x_{m,c} \leq fit_{m,c} \qquad \forall m \ \forall c \qquad \text{(III.3)}$$

- capacity: since penalties apply only if there is a positive shortfall, the objective function would be non-linear. To avoid that, we assume we will never oversupply Marines to schools, resulting in:

$$\sum_{m \in \mathcal{M}} x_{m,c} \leq demand_c \qquad \forall c \tag{III.4}$$

# IV. A DECISION SUPPORT SYSTEM FOR RECRUIT DISTRIBUTION

## A. ARCHITECTURE AND IMPLEMENTATION

Following is a discussion of the RDdss architecture and its implementation. First, the architectural components are discussed. Then, the steps taken to build the RDdss are described.

### 1. Architecture

The architecture of the RDdss is depicted in Figure 7. This illustration shows how the six major components of the system are related. We will describe this architecture by examining each component in the following order: switchboard, relational database, preprocessor, assignment model, solver, and analyzer.

The switchboard is one of two components providing an interface between the user and the RDdss. It is the mechanism by which the user controls the operation of the system. Figure 8 illustrates the main switchboard of the RDdss. It is displayed

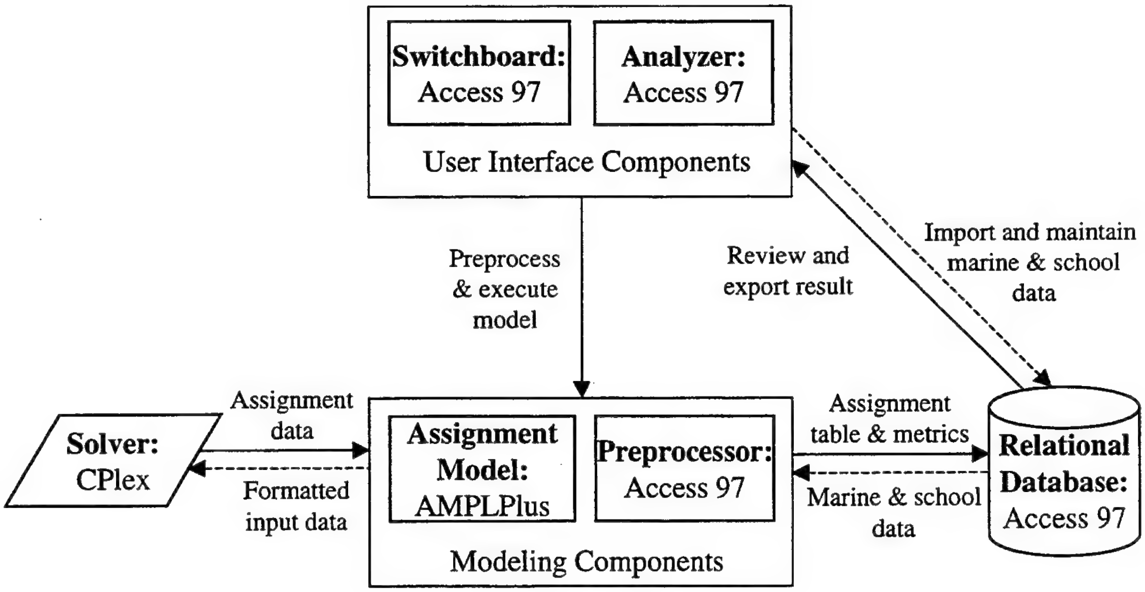


Figure 7. RDdss Architecture

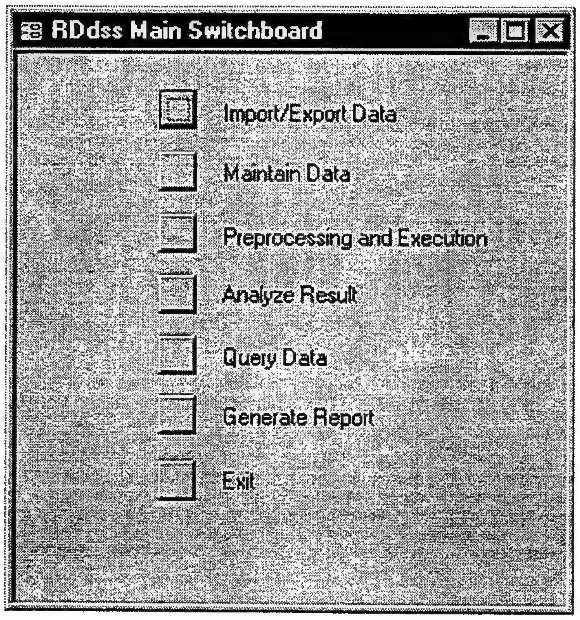


Figure 8: RDdss Main Switchboard

when the application is started. As this illustration depicts, the user can perform the following functions by simply pointing and clicking on the appropriate button: import and export data, maintain data, preprocessing and execution of the assignment model, analyze results, query data, and generate reports. All the RDdss switchboards were generated using Access 97.

The relational database technology was used in the RDdss [Ref. 8]. The actual table relationships are shown in Figure 9. These tables were developed in Access 97, which uses the relational technology. In addition to a few flat files used to communicate with the assignment model, all the data for the RDdss is stored in this database.

The preprocessor is one of the two modeling components. It consists of Access 97 Visual Basic for Applications (VBA) code that computes information necessary for the assignment model [Ref. 9]. It determines the demand of each class, the appropriate penalty associated with each class, and the Marine-to-class fitness matrix. The code for these assignment model inputs is found in Appendix D, lines 452-516 and 4305-4812.

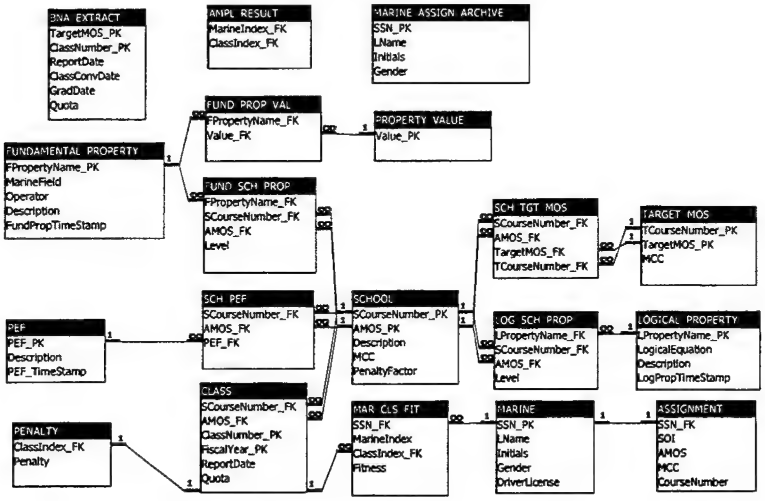
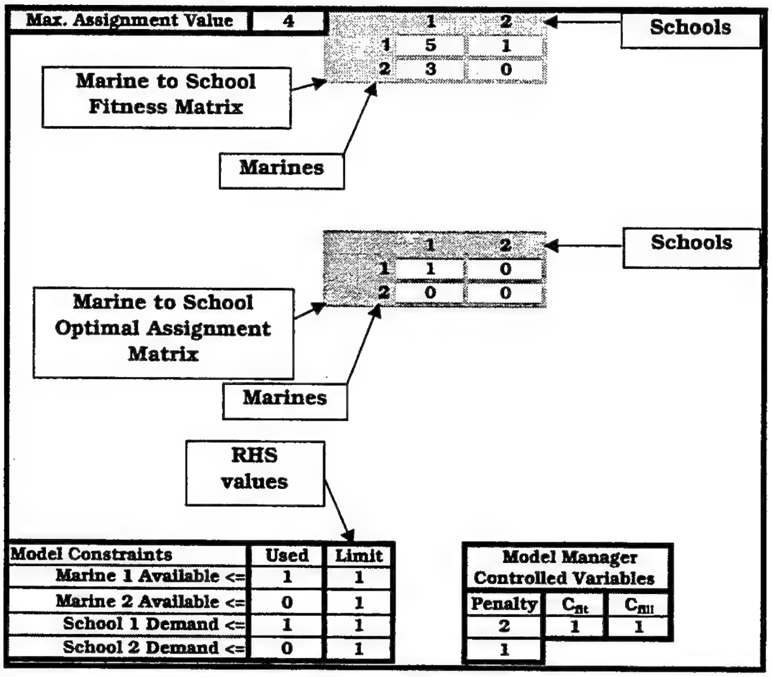


Figure 9. RDdss Relational Schema

The assignment model is the other component of the modeling components. It has been modeled in AMPL [Ref. 10], a language and a computing environment for expressing, solving, and analyzing mathematical programming problems (minimizing or maximizing a function of decision variables, subject to constraints on the variables). A complete discussion of the assignment model's math model is available in Chapter III, Section D.

The next component is the solver. It communicates directly with AMPL, the modeling language of AMPLPlus. As mentioned in Chapter II, it uses an algorithm called CPlex, which is an optimization package for linear, network and integer programming. Working in conjunction with AMPLPlus, the solver finds an optimal solution.

The final component is the analyzer. It is the other component making up the user interface. This is where the RDdss manager analyzes assignment results, seeking insights for developing a "good" solution. Insight is developed mostly by the graphs and numerical information available in the analyzer. These were generated in Access 97 using its report generator and VBA code. The programming used in the analyzer is shown in Appendix D and includes lines 1-310, 2156-2214, and 3756-3827.



| Max. Assignment Value | 4 |
|---|---|

|  | 1 | 2 |
|---|---|---|
| 1 | 5 | 1 |
| 2 | 3 | 0 |

|  | 1 | 2 |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 0 | 0 |

| Model Constraints | Used | Limit |
|---|---|---|
| Marine 1 Available <= | 1 | 1 |
| Marine 2 Available <= | 0 | 1 |
| School 1 Demand <= | 1 | 1 |
| School 2 Demand <= | 0 | 1 |

| Model Manager Controlled Variables |  |  |
|---|---|---|
| Penalty | $C_{nt}$ | $C_{nll}$ |
| 2 | 1 | 1 |
| 1 |  |  |

Schools
Marine to School Fitness Matrix
Marines
Schools
Marine to School Optimal Assignment Matrix
Marines
RHS values

Figure 10. RDdss Demonstration Solver

## 2. Implementation

Building the RDdss was an iterative process. Some steps took numerous iterations, and some were straightforward. Below is a listing of the 14 major steps it took to build the RDdss.

1. Analyzing the current system was the first step. This included 1) interviewing USMC and DSAI personnel, 2) reviewing available documentation, and 3) operating the RDM.

2. After gathering the current system data, it was necessary to develop a logical understanding of the system. This was accomplished by modeling it with BPWin. Appendix B is the result of this step.

3. With this system understanding, it was possible to envision how the new system would make assignments. Using standard optimization techniques [Ref. 11], a small-scale demonstration solver was built using Excel 97. It is shown in Figure 10.

4. Combining the system understanding and the data requirements for the demonstration solver, it was possible to envision a data structure for the reengineered system. This lead to the development of a third normal form relational schema for the RDdss [Ref. 3].

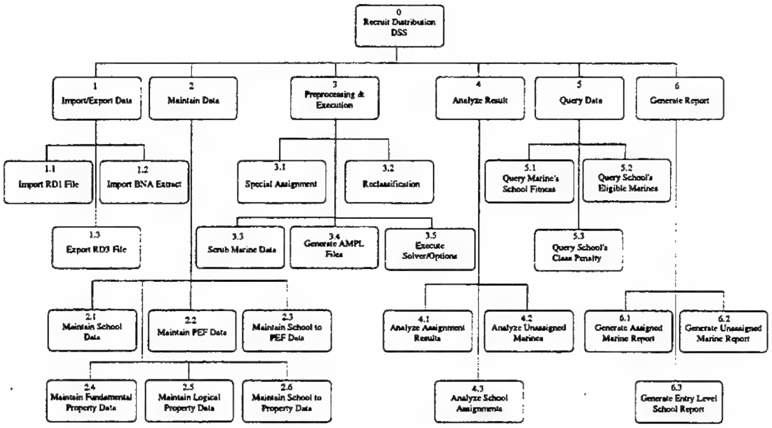

0
Recruit Distribution DSS
1
Import/Export Data
2
Maintain Data
3
Preprocessing & Execution
4
Analyze Result
5
Query Data
6
Generate Report
1.1
Import RD1 File
1.2
Import BNA Extract
1.3
Export RD3 File
3.1
Special Assignment
3.2
Reclassification
3.3
Scrub Marine Data
3.4
Generate AMPL Files
3.5
Execute Solver/Options
5.1
Query Marine's School Fitness
5.2
Query School's Eligible Marines
5.3
Query School's Class Penalty
2.1
Maintain School Data
2.2
Maintain PEF Data
2.3
Maintain School to PEF Data
4.1
Analyze Assignment Results
4.2
Analyze Unassigned Marines
6.1
Generate Assigned Marine Report
6.2
Generate Unassigned Marine Report
2.4
Maintain Fundamental Property Data
2.5
Maintain Logical Property Data
2.6
Maintain School to Property Data
4.3
Analyze School Assignments
6.3
Generate Entry Level School Report

Figure 11. RDdss Decomposition Diagram

5. Using the relational schema as a model, the RDdss tables were built in Access 97. The results of this were seen earlier in Figure 9.

6. As the vision for the reengineered system came more into focus, an intuitive navigational scheme was conceived. This lead to the development of a decomposition diagram for the RDdss. It is demonstrated in Figure 11.

7. Using the decomposition diagram as a model, the user interface was built for the RDdss in Access 97. The Main Switchboard was seen earlier in Figure 8.

8. With the switchboards now in place, VBA coding was commenced [Ref. 9]. This made each switchboard functional and useful.

9. Building upon the concept of the demonstration solver from step 3, a formal mathematical assignment model for the RDdss was developed. This was introduced in Chapter III, Section D.

10. The COTS applications, AMPLPlus and CPlex were installed on the personal computer containing the RDdss.

11. With AMPLPlus now installed, it was possible to code the assignment model into the application. This made it possible to verify our design.

12. Manually creating input files for AMPLPlus is time consuming. So, the preprocessor was coded. This automated the generation of the class demand, class penalty, and Marine-to-class fitness matrix.

13. Further automation of the RDdss was accomplished by interfacing Access 97 and AMPLPlus. This required both VBA and AMPL coding. The end result was a seamless operation of the two applications.

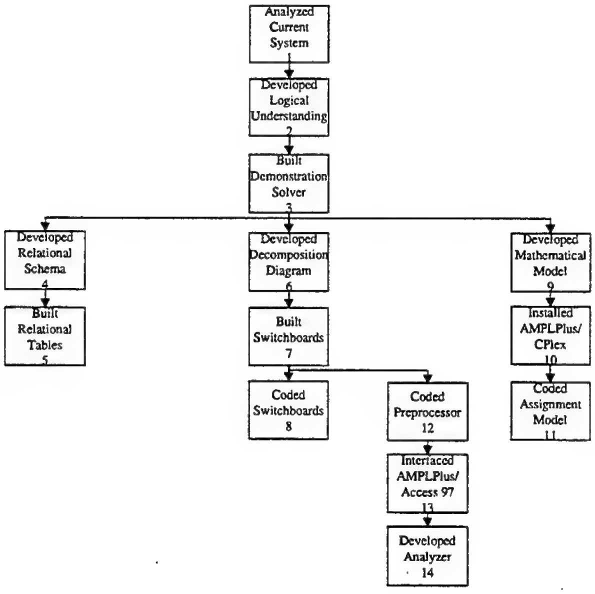

Analyzed Current System
Developed Logical Understanding 2
Built Demonstration Solver 3
Developed Relational Schema 4
Built Relational Tables 5
Developed Decomposition Diagram 6
Built Switchboards 7
Coded Switchboards 8
Developed Mathematical Model 9
Installed AMPLPlus/ CPlex 10
Coded Assignment Model 11
Coded Preprocessor 12
Interfaced AMPLPlus/ Access 97 13
Developed Analyzer 14

Figure 12. Actual RDdss Build Sequence

14. Finally, much thought was given to ways of gaining insight into the assignment results. This lead to the development of the analyzer, which was discussed briefly in the previous section.

Some of these steps were performed in parallel. The actual order followed in building the RDdss are depicted in Figure 12.

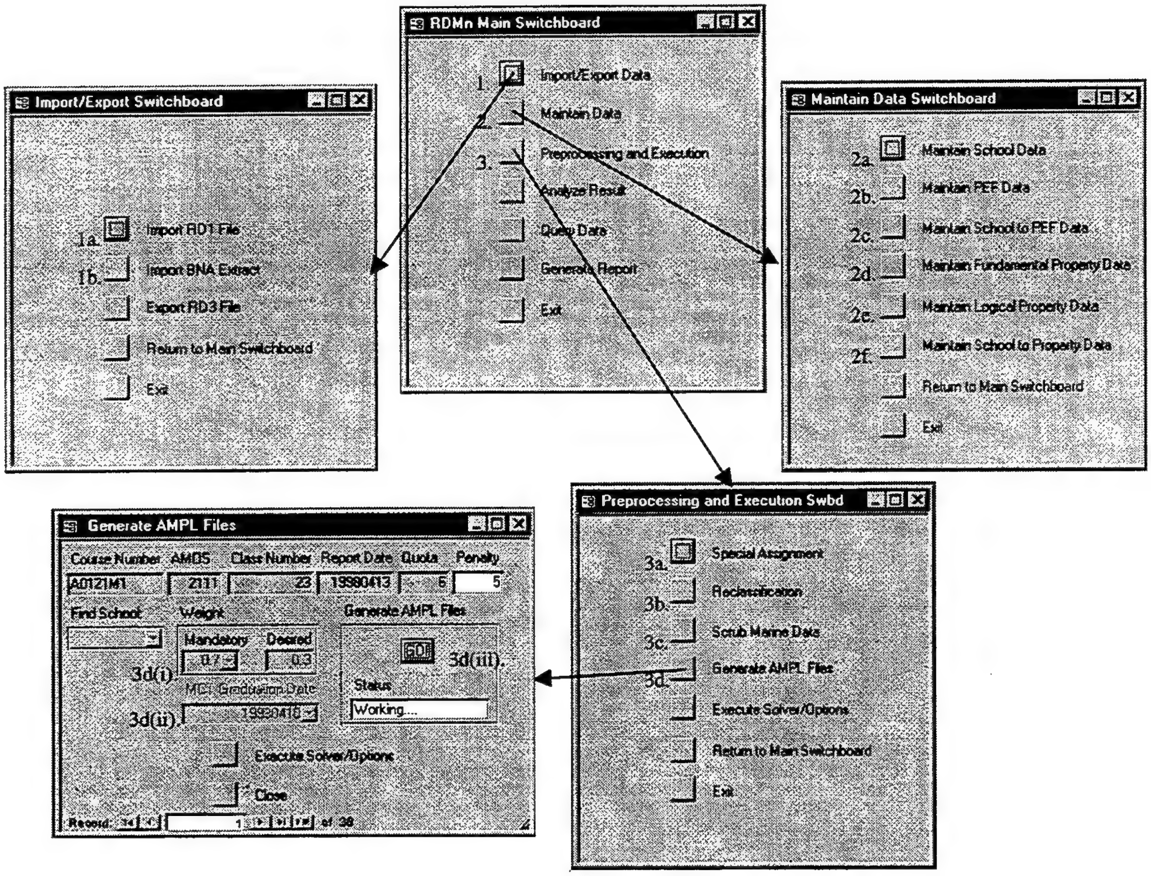
## B. USING THE RECRUIT DISTRIBUTION DECISION SUPPORT SYSTEM

In this section, we will discuss using the RDdss. Three aspects are covered, 1) setting up a run, 2) model execution, and 3) customizing a run and results. Each is discussed in the order given here.

### 1. Setting up a Run

Setting up a run of the RDdss is straightforward and easy to do. Figure 13 graphically illustrates the steps involved. Starting with step 1 on the graph and working through to step 3d(iii), we will discuss a run set-up.

The RDdss manager starts at the top of the Main Switchboard. Here, he or



| Course Number | AMOS | Class Number | Report Date | Quota | Penalty |
|---|---|---|---|---|---|
| A0121M1 | 2111 | 23 | 19980413 | 6 | 5 |

Mandatory: 0.7
Desired: 0.3
MCT Graduation Date: 19980410
Status: Working....
Record: 1 of 38

Figure 13. Setting up a Model Run

she clicks on the Import/Export Data button. This will cause the Main Switchboard to close and the Import/Export Switchboard to open. From this switchboard, the RDdss manager will import the RD1 file and the BNA extract into the RDdss MODS. This is easily accomplished by clicking on the corresponding buttons for importing these files. Finally, the RDdss manager will click on the "Return to Main Switchboard" button, causing the Import/Export Switchboard to close and the Main Switchboard to reopen.

The next step is data maintenance. This includes updating, editing, and/or deleting of school related information. The RDdss manager gets to the Maintain Data Switchboard by clicking on the appropriate button on the Main Switchboard.

From the Maintain Data Switchboard, the RDdss manager will start at the top and work his or her way down to the bottom. The various buttons on this switchboard open forms where data maintenance is performed. For instance, the first button opens the Maintain School Data form, allowing the RDdss manager to create, edit, and/or delete a school. The other buttons allow similar operations with the PEFs, and fundamental and logical properties. After the RDdss manager has finished maintaining the data, he or she returns to the Main Switchboard by clicking on the "Return to Main Switchboard" button.

After data maintenance, the RDdss manager is ready to perform data preprocessing in prepartion for a model run. By clicking on the Preprocessing and Execution button, the Main Switchboard closes and the Preprocessing and Execution Switchboard opens. From here, as on all other switchboards, the RDdss manager starts at the top and works down.

The first three preprocessing buttons directly effect the generation of the class demand, class penalty, and Marine-to-class fitness matrix. This is why they come before the fourth button, Generate AMPL Files. Preprocessing in the correct order will reduce the amount of time spent setting up the run, by minimizing the need to generate a new class demand, class penalty, and Marine-to-class fitness matrix.

The first preprocessing button concerns special assignments. Specially assigning a recruit affects the AMPL input in two ways. First, the respective member is not included in the Marine-to-class fitness matrix, since they already have an assignment. Second, the class demand for the class the recruit is assigned to must decrease by one. Otherwise, it may become overfilled.

The second preprocessing button concerns reclassification of a Marine. Reclassified Marines affect the AMPL input in two ways. First, the class demand of the class they are no longer assigned to must increase by one. This provides an opportunity to fill the empty seat during a subsequent run. Second, the class demand of the class the Marine has been reassigned to must decrease by one. Otherwise, it may become overfilled.

The third preprocessing button concerns data scrubbing, which also affects the AMPL input. It is an attempt to identify and correct errors relating to the Marines. An example is a Marine who has been given an unknown PEF. Since it is not identified by the RDdss, the member will receive a zero fitness score for every class. This will prevent the Marine from getting an assignment.

Following the completion of special assignments, reclassifications and data scrubbing, the RDdss manager is ready to generate the AMPL input files. After going to the Generate AMPL Files form, the first concern is specifying the weights for the mandatory and desired properties. This is specified by the RDdss manager at step 3d(i). What this does is indicate how important the mandatory properties are with respect to the desired properties. For instance, if the RDdss manager decided to give a weight of 1.0 to the Mandatory selection, the Desired selection would automatically fill in with 0.0. This would indicate the desired properties are of no consequence. The default for these two selections is 0.7 for Mandatory, and 0.3 for Desired.

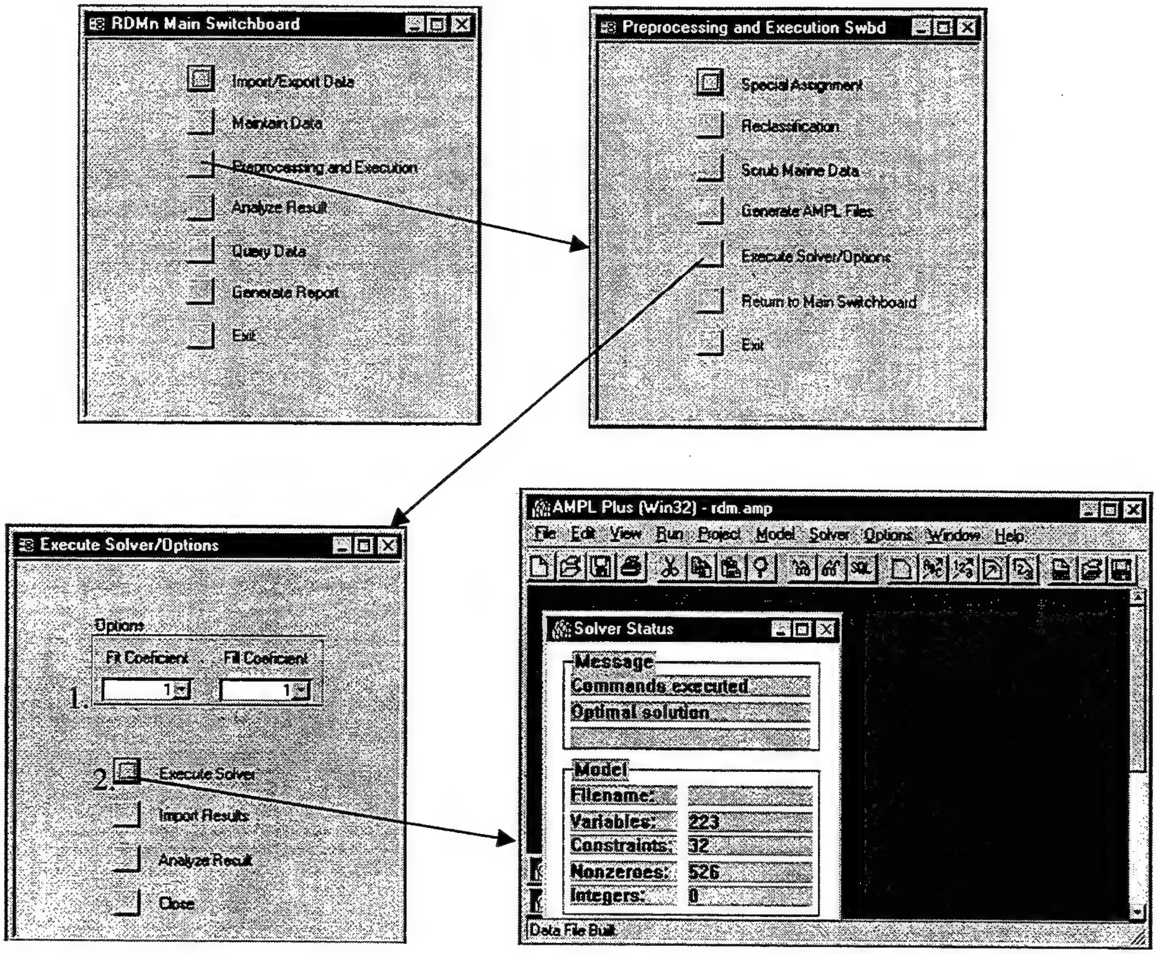
Next, at step 3d(ii), the RDdss manager selects the MCT graduation date desired. This is selected from the drop down list. Then, the RDdss manager pushes the "GO!" button. The system will now generate the AMPL input files, completing the run set-up.

## 2. Model Execution

Execution of the assignment model takes place after the run set-up has been completed. Figure 14 graphically illustrates the steps taken to execute the model. A discussion of this is now given.

From the Main Switchboard, the RDdss manager points and clicks on the Preprocessing and Execution button. This takes him or her to the corresponding switchboard. From here, the Execute Solver/Options selection is made. This opens up the form where the model is executed from.

The Execute Solver/Options form provides the RDdss manager the ability to set the fit and fill coefficients for the objective function. The default values are 1 for



Fit Coeficient: 1
Fill Coeficient: 1

Message
Commands executed
Optimal solution

Model
Filename:
Variables: 223
Constraints: 32
Nonzeroes: 526
Integers: 0

Figure 14. Model Execution

each coefficient. The RDdss manager can change either of these with a whole number from 0 - 100. The significance of these numbers is discussed in detail later in this chapter. Once the desired coefficients have been chosen, the Execute Solver button is pushed.

By selecting the Execute Solver button, AMPLPlus is launched. This application will run without user interaction. First, it will import the RDdss mathematical assignment model. Next, the specified fit and fill coefficients are retrieved. Finally, the preprocessed data generated during the run set-up are imported. Using this input data, AMPLPlus works in conjunction with the solver and generates an optimal assignment solution.

With the generation of an optimal assignment solution, the results are sent to an output file called "rdm.out." AMPLPlus, having completed its task, now closes. This indicates an optimal solution was found and is available for importing. So, the RDdss manager now imports the results by pushing the Import Results button on the Execute Solver/Option form.

### 3. Customizing a Run and Results

As mentioned in the previous subsection, the RDdss manager can set the fit and fill coefficients of the objective function. This is a key capability of the new system. It is the means by which the RDdss manager can customize a run and "game" the RDdss. This "gaming" of the new system is essential for determining a "good" assignment result.

For instance, if the fit coefficient is set to 0, and the fill coefficient is set to a number $> 0$, say 1, the assignment model will maximize the fill of schools having early report dates. This has both advantages and disadvantages. One advantage is the average wait time for Marines to attend a school is minimized. Another is that the number of unfilled school seats, having a report date between the current and next scheduled model run, is minimized. However, a disadvantage is the average Marine-to-school fitness score is guaranteed to be the lowest score of all runs.

Conversely, if the fit coefficient is set to a number $> 0$, say 1, and the fill coefficient is set to 0, the assignment model will maximize the average Marine-to-school fitness. An advantage here is the increased chance of each Marine successfully completing their ELS, establishing a pattern of success early in theri career. However, one disadvantage is the wait time for a Marine to attend a school is guaranteed to be the longest. Also, the number of unfilled school seats, having a report date between the current and next scheduled model run, is guaranteed to be the greatest.

Neither of the above solutions is ideal. A better solution is some where between these two extremes. By changing the fit and fill coefficients of the objective function, the RDdss manager has a means for customizing each run to ensure a "good" assignment result is found.

## C. INNOVATIONS IN USER INTERACTION

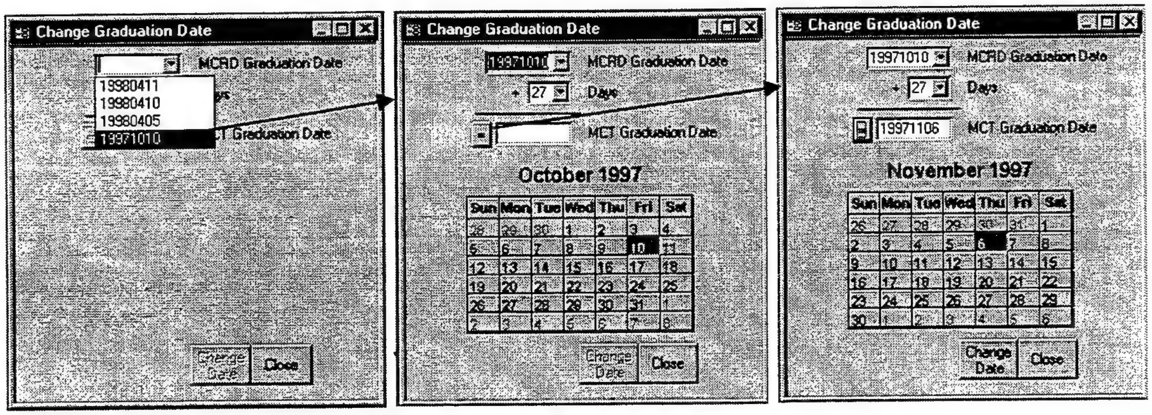
A number of user interaction innovations have been made in the RDdss. Two of them have already been discussed. These were the intuitive navigational scheme using the switchboards, and the elimination of manually entering data already available in the system. One more note worthy innovation is now introduced. It is a date change innovation.

The date change innovation concerns changing the graduation date of the Marines in the downloaded RD1 file. This is necessary for generating correct assignment results. The date in the downloaded RD1 file is based on graduation from the MCRD. However, the model needs to know when the Marines are available for assignment. Availability is based on the MCT graduation date.

The old system solution to this problem took three steps. First, the RDM manager used a calendar to determine the MCT graduation date. This is normally 27 days following completion of MCRD, and is always on a friday. So, the RDM manager will find the MCRD graduation date on the calendar and count 27 days. The friday nearest to this day is the day desired. Second, this MCT graduation date, along with the RD1 file, is sent to another branch in the USMC where the date change is made. Third, the file is returned to the RDM manager, who imports the file into the RDM.

The reengineered system has streamlined this process. Within the RDdss, the RDdss manager makes the date change. It is accomplished with the push of a couple buttons.

Figure 15 shows how these changes are made. First, the RDdss manager selects the MCRD graduation date requiring a change. Once this is selected, a calendar appears. Its purpose is to help the RDdss manager correctly identify the MCT graduation date. Next, the "=" button is pushed. Its purpose is to add the number of



Change Graduation Date

MCRD Graduation Date

19980411
19980410
19980405
19971010

Change Graduation Date

MCRD Graduation Date: 19971010

+ Days: 27

MCT Graduation Date:

October 1997

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| 28 | 29 | 30 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Change Graduation Date

MCRD Graduation Date: 19971010

+ Days: 27

MCT Graduation Date: 19971106

November 1997

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| 26 | 27 | 28 | 29 | 30 | 31 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | 1 | 2 | 3 | 4 | 5 | 6 |

Figure 15. Changing a Graduation Date

days specified in the "Days" field (default is 27), to the MCRD graduation date. This updates the calender as shown on the far right. When the correct date is found, the "Change Date" button is pushed. This makes the desired change to the data in the RDdss database.

## D. INNOVATIONS IN ANALYZING MODEL RESULTS

As discussed earlier, the RDdss manager has the capability of "gaming" the system by using different fit and fill coefficients. However, having this ability is not enough. The RDdss manager still needs a means to compare one run against another.

It is hard, if not impossible, to estimate the quality of a solution simply by looking at a list of assignments. The actual assignment results depend on the values of the fit and fill coefficients, which cannot be set at the ideal levels without understanding the impact on solution quality. So, analyzing run results to get insight and develop a "good" solution is now discussed.

Four measures are considered particularly useful in evaluating alternative solutions.

1. Fitness premium. The Marine-to-school fitness function is defined in a way that for each school, the average fitness over all Marines *eligible* for that school is 100. So, for any school, if the average fitness computed over Marines *assigned*

to that school is greater than (or less than) 100, the difference represents a positive (or negative) premium.

2. Unfilled seats in first week. Since the model is run nearly every week, seats left unfilled in classes starting in the first week will remain unfilled. This represents wasted resources.

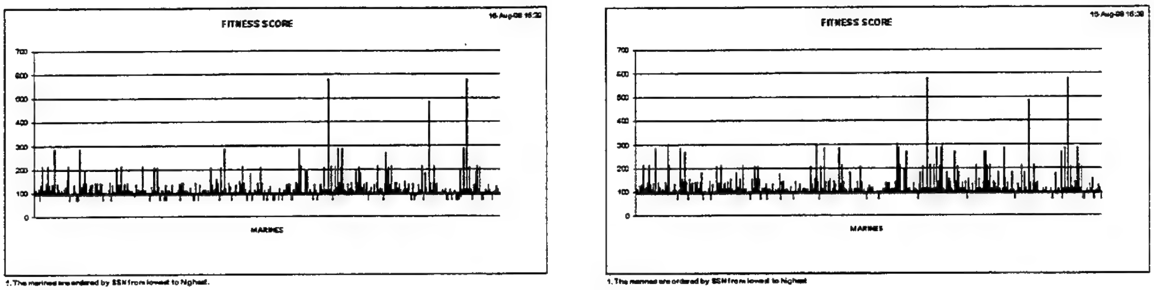
3. Average wait time for Marines. Experience has shown that the longer a Marine waits before attending their ELS, the greater the chance of disciplinary problems. Therefore, it is generally better to keep this wait period short.

4. Unassigned Marines. A purpose of the system is to assign each Marine to a school. Therefore, knowing the number of unassigned Marines is an important measure.

To provide the RDdss manager insight into the assignment results, these measures have been incorporated into output representations, including summary statistics and detailed graphs. Each output provides both numerical and visual information. The numbers are used for quantitative comparisons. The graphical information provides a big picture view for use in a qualitative comparison.

As an example, consider the two fitness graphs in Figure 16. Both of these were produced using an RD1 file of 344 Marines and an BNA extract with school data covering a 120 day period. Below each graph is amplifying and numerical information. The bar chart on the left has a fit coefficient of 0 and a fill coefficient of 1. Additionally, its fitness premium is 28 (=128-100). The bar chart on the right has a coefficient of 1 for both fit and fill. Its fitness premium is 38 (=138-100). The fit numbers are quantitatively comparable. The fitness premium for the run on the right is 36% (=100% x [38-28]/28) better than the run on the left.

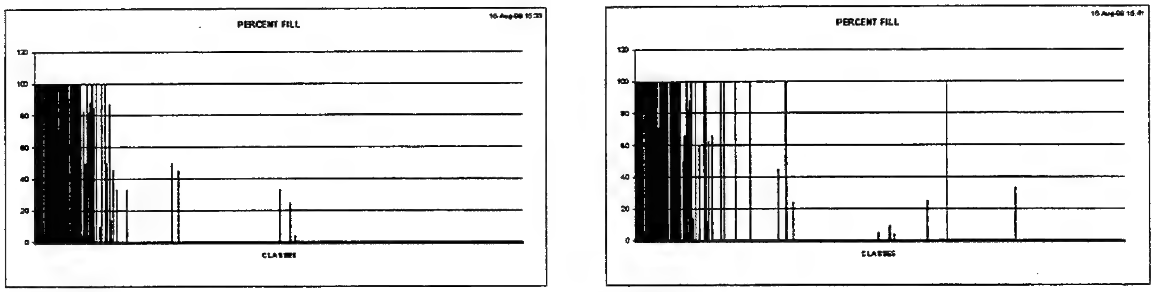
The above numerical analysis is also supported by visually comparing the bar charts. Notice the large number of blips falling below the fitness score of 100 on the left graph. Each blip represents a Marine, and there are actually 44 of them. Conversely, notice the comparatively small number of blips below 100 on the right graph. There are 21 of them. The visual comparison of the two graphs reveals the



1. The marines are ordered by SSN from lowest to highest.
2. Fit coefficient = 0
3. Fill coefficient = 1
4. Average fitness = 129
5. Fitness premium = 28

1. The marines are ordered by SSN from lowest to highest
2. Fit coefficient = 1
3. Fill coefficient = 1
4. Average fitness = 129
6. Fitness premium = 38

Figure 16. Two alternative results: Fit



1. The classes are ordered by penalty from highest to lowest.
2. Fit coefficient = 0
3. Fill coefficient = 1
4. Marines assigned = 341
5. Average wait (weeks) = 1.20

1. The classes are ordered by penalty from highest to lowest.
2. Fit coefficient = 1
3. Fill coefficient = 1
4. Marines assigned = 341
5 Average wait (weeks) = 1.80

Figure 17. Two alternative results: Fill

qualitative difference. The graph on the right indicates a higher quality Marine-to-school fit. Of course, this supports the numerical comparison performed earlier.

As another example, consider the two fill graphs in Figure 17. These are based on the two model runs just described. In other words, the left graph has a fit coefficient of 0 and a fill coefficient of 1, and the right graph has a coefficient of 1 for both fit and fill.

Visually, these two graphs display an obvious difference. To appreciate how they are different requires an understanding of the significance of the blips. As note 1 of each graph indicates, "The classes are ordered by penalty from highest to lowest." Recall from Chapter III, this means the ordering is by class report date, starting with the earliest date. In other words, the bar chart on the left indicates almost every Marine will start school within the first three weeks. As discussed earlier, this is desirable. The bar chart on the right has its blips more spread out. So, by visually comparing the two graphs, the left one has a higher quality with respect to minimizing wait time.

This is easily validated by the numbers given in note 5 of each graph. The average wait of a Marine from the left bar chart is 1.33 weeks, and the average wait from the other bar chart is 1.69 weeks. Now we quantitatively compare these values. The results from the left graph are 27% (=100% x [1.69-1.33]/1.33) better than the right.

In the following table, the results from three runs are given. The measures on the left are the four measures described earlier. Each run used the same 344 Marines and 120 days worth of school classes. The information for the first and third runs is from the two model runs discussed above. The second run was determined by "gaming" the system. In other words, we tried numerous combinations of the fit and fill coefficients until a "good" solution was found. Run 2 was the result. Understand though, we are not saying "best" solution. Only the USMC can determine what is best for them.

| | Run 1<br>(fit=0; fill=1) | Run 2<br>(fit=1; fill=11) | Run 3<br>(fit=1; fill=1) |
|---|---|---|---|
| Unfilled Seats (1st week) | 23 | 23 | 23 |
| Unassigned Marines | 3 | 3 | 3 |
| Average Wait (weeks) | 1.33 | 1.33 | 1.69 |
| Fitness Premium | 28 | 31 | 38 |

The reason we feel Run 2 is a "good" solution is based on a quantitative comparison of the measures and one additional piece of information. It is the fact the USMC feels it is more important to minimize wait time than to maximize the Marine-to-school fit. The remainder of this section explains how we decided Run 2 was a "good" solution.

Starting with the values for unfilled seats, it is obvious all three runs are equal. The same is true for the unassigned Marines measure. So, these two measures provide no insight for comparison purposes.

Concerning the average wait, the values of Runs 1 and 2 are equal. Further, their 1.33 value is numerically smaller than the 1.69 value of Run 3. This is good for Runs 1 and 2. As was proven earlier, a value of 1.33 is 27% better than 1.69.

Finally, there is the fitness premium measure. The first observation made is that Run 1 is a candidate for elimination. The reason is as follows. Run 2 has the same score as Run 1 for the first three measures in the table, but it shows an 11% (=100%x[31-28]/28) increase in the fitness premium. So, Run 1 is eliminated from the competition.

This leaves the last two runs. Run 3 shows a 23% (=100%x[38-31]/31) increase in fitness premium over Run 2. This creates a conflict. Run 3 has a 27% increase in average wait time, which is bad. Conversly, it has a 23% increase in fitness premium, which is good. To decide which solution is better, more information is necessary. This comes from the fact mentioned earlier, that the USMC feels it is more important to minimize wait time than to maximize the Marine-to-school fit.

We can now conclude Run 2 is a better solution than Run 3. Here is why. Run 2 minimizes average wait by 27% by giving up a 23% increase in Marine-to-school fit. This agrees with the USMC desire to more strongly emphasize a minimum waiting time. So, Run 2 is a better solution than Run 3. Since it also satisfies the known desires of the USMC, we feel it is a "good" solution as well.

## E. OBJECTIVE COMPARISON OF OLD AND NEW SYSTEM SOLUTIONS

In the previous section, we described how a set of solutions generated by the RDdss are compared to determine a "good" solution. Using the same approach, we now demonstrate how to objectively compare the results from the old system with the new system. This provides the USMC a means of determining which system's solution is better, if either. Comparison of the new and old system results has been made easy by automating the importing of the RDM result into the RDdss.

We have summarized the important data for the old and new systems in the following table, based on the same 344 Marines and 120 days of school classes discussed earlier. Since the assignment algorithm for the RDM was designed to maximize fill, a run from the RDdss based strictly on fill has also been included. It is the run with a fit coefficient of 0, and a fill coefficient of 1. The last column, labeled "Run 2," is a "good" solution we found by gaming the RDdss.

| | Run 1<br>(fit=0; fill=1) | RDM | Run 2<br>(fit=1; fill=10) |
|---|---|---|---|
| Unfilled Seats (1st week) | 23 | 23 | 23 |
| Unassigned Marines | 3 | 3 | 3 |
| Average Wait (weeks) | 1.33 | 1.34 | 1.34 |
| Fitness Premium | 28 | 29 | 32 |

We start by objectively comparing the RDdss fill-based result with the RDM result. They are labeled "Run 1" and "RDM," respectively. As we shall prove, the RDdss has done a better job at filling early school seats than the RDM.

Of the four measures, only one provides any useful insight for this comparison. It is the average wait measure. The reason the other three do not apply is as follows. The first two measures have the same values, so they provide no insight. The last measure is not too meaningful, since Run 1 is strictly based on maximizing fill. In other words, it disregards fit because its fit coefficient is set to 0. This leaves the average wait measure.

A quantitative comparison of the average wait shows the RDdss solution is 1% (100%x[1.34-1.33]/1.33) better than the RDM solution. This is explainable by considering the approach used in each model. The RDM assignment algorithm maximizes the fill of early school seats by looking at each school in a sequential fashion. In contrast, the RDdss conducts a global comparison of Marines and schools. In other words, it goes through numerous iterations until it finds an optimal fill solution. This show that the RDM assignment algorithm is only closely approximating the optimized assignment result.

Finally, we "gamed" the new system until a "good" solution was found. This is Run 2. It has a fit coefficient of 1 and a fill coefficient of 10.

The objective comparison of the RDM and Run 2 solutions is straigtforward. Since the first three measures are the same for both results, they provide no insights. This leaves only the fitness premium measure. A quantitative comparison of this measure shows Run 2 has a 10% (100%x[32-29]/29) better solution than RDM. This shows the benefit of "gaming" the RDdss.

## F. SUMMARY

This relatively long chapter covered some important points related specifically to the RDdss. The architectural components of the new system were discussed first, followed by the steps taken to actually build this DSS. Then, from an operational point of view, using the RDdss was discussed. This included setting up a run, executing the model, and customizing a run for determining a "good" solution. Next, two innovations were discussed in great detail. These included the date change innovation and the analyzing model results innovation. Finally, a means of objectively comparing the solution from the old and new systems was discussed.

We truly believe the innovations and capabilites introduced by this prototype can greatly benefit the USMC's Marine Enlisted Assignments branch. It can help the USMC to more effectively accomplish its mission as stated by the Commandant of the Marine Corp, "to put the right Marine in the right place at the right time with the right skills and quality of life."

# V. CONCLUSION

This concluding chapter has three short sections, followed by a final comment. In the first section, we reiterate the significant contributions made in the thesis. This is followed by a brief description of some lessons we learned. Then, some recommended improvements for the prototype are given. Following this last section, we comment on how this thesis work might benefit the other branches of the United States military.

## A. SIGNIFICANT CONTRIBUTIONS

Four significant contributions beneficial to the USMC were made in this thesis. These were discussed in detail in the previous three chapters. A list and summary of the contributions is given here.

- Analysis and articulation of the recruit distribution process - by interviewing USMC and DSAI personnel, reviewing available documentation, and operating the RDM we were able to articulate an understanding of the recruit distribution process using IDEF process modeling. Additional data modeling was articulated in a third normal form relational schema.
- Development of a mathematical model - by analyzing the assignment process, criteria, and constraints, USMC policies and objectives, a mathematical model for the new system was developed.
- Fully functional prototype - an intuitive and easy to understand new system that seamlessly interfaces with the old system was built. It provides an objective means of comparing assignment results of both systems, and was developed using COTS software applications.
- Elimination of the proprietary solver and associated contractor lock-in - this was accomplished by replacing the proprietary solver with two COTS applications.

## B. LESSONS LEARNED

Our original goal was to build an As-Is IDEF0 model of the RDM, about two months after starting the thesis. This worked out fine. The next goal was to take another six weeks to develop a To-Be IDEF0 model [Ref. 1]. This did not work out, for two reasons. First, our understanding of the entire recruit distribution process was still not mature enough to build the To-Be model. Nearly half a year was spent just thinking about its many details. Second, it is nearly impossible to envision the new system without first knowing the capabilities and limitations of the application used to build the prototype. One extreme envisions the impossible, while the other barely touches the reengineered system's potential. *The lesson learned from this was that properly reengineering a process requires two things. A detailed understanding of the current process environment, and a good working knowledge of the capabilities and limitations of the application used to build the prototype.*

Getting feedback from the Marine Enlisted Assignments branch was not always easy. Correspondence normally occured by e-mail. There was an inverse relationship between the number of questions asked and the number of answers given. As the one increased, the other decreased. *So, another lesson learned was that it is best to limit each correspondance to a question or two.*

While writing code, it is best to simultaneously document the code's purpose. Going back and trying to provide documentation at the end is too frustrating. This requires a considerable amount of discipline, but is well worth the effort. *The lesson learned here is document as you code.*

## C. PROTOTYPE IMPROVEMENTS

A big part of this thesis involved the building and refinement of the prototype. The following topics are areas where the RDdss could use improvements.

### 1. Speed Improvements

#### *a. Preprocessing Data*

Of all the steps necessary to complete a run of the RDdss, the preprocessing step is the most time consuming. For the 344 Marines and approximately 500 school classes analyzed in this thesis, it takes about 30 minutes to preprocess this data. This is about ten times longer than the assignment model takes to find an optimal solution. As the size of the preprocessing data is doubled, so is the time needed to preprocess it. Finding a way to speed up this process is desirable. One possibility is to find a way to perform all the preprocessing in main memory, with only two accesses to the hard drive. The first access is to get the necessary Marine and school data. The second is to write the preprocessed results back to the hard drive.

#### *b. Analyzing Result*

Compared to all other forms and switchboards in the RDdss, opening the "analyze results" forms is the most time intensive. With the 344 Marines and approximately 500 school classes, it takes just under a minute for any of these three to open. This feels like an eternity when you are in a hurry. The reason it takes so long is the creation of temporary tables, numerous calculations and comparisons, and retrieving of data from flat files. Finding a way to reduce this time is desirable. One possibility is to create another form which would contain only the four objective measures. Currently, the RDdss manager must go to two different forms for this information. Each containing information and displays in addition to the objective measures, which slow them down.

### 2. Multiple Solution Storage

Currently, the prototype will only save the results of one solution. This means the RDdss manager must either print-out hard copies of the graphs, or manually write down each solution's objective measures for comparison with other runs. An improvement here should allow the storage of at least three solutions. This would permit the RDdss manager to easily compare each solution. He or she could then eliminate one or two undesirable solutions, while continuing to search for a "good" one.

### 3. Administration Switchboard

To look at all the schools or properties in one display, the RDdss manager must open up the associated school or property table. A reason for doing this is system administration. For instance, if a supervisor is auditing the RDdss's property data, looking at all the data at one time is convenient. A simple way of including this feature is by adding another switchboard. The switchboard's buttons would have VBA code associated with them, causing the desired tables to open.

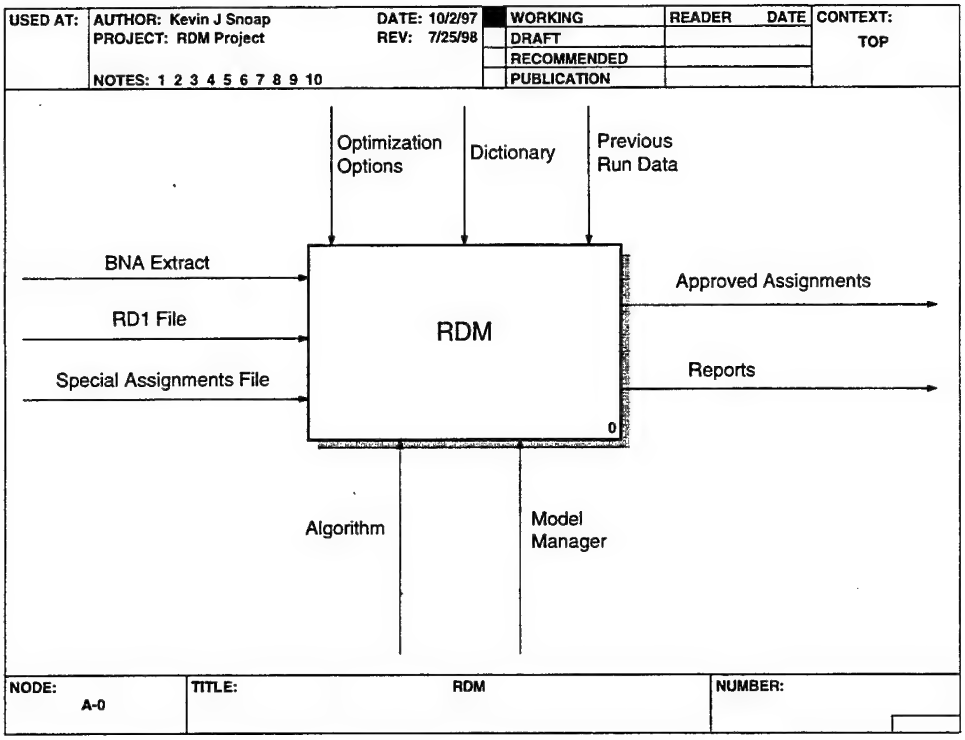
Any benefits derived from this thesis work were specifically aimed at the USMC. However, it does not have to end there. Like the USMC, other branches of the United States military must deal with the problem of how best to assign their service members to schools. Since all the services require their enlisted personnel to take the ASVAB test, the approach used in our prototype might prove an ideal solution. Further, our efforts may also contribute to the reengineering of other assignment models. So, this thesis work could actually benefit all branches of the United States military.

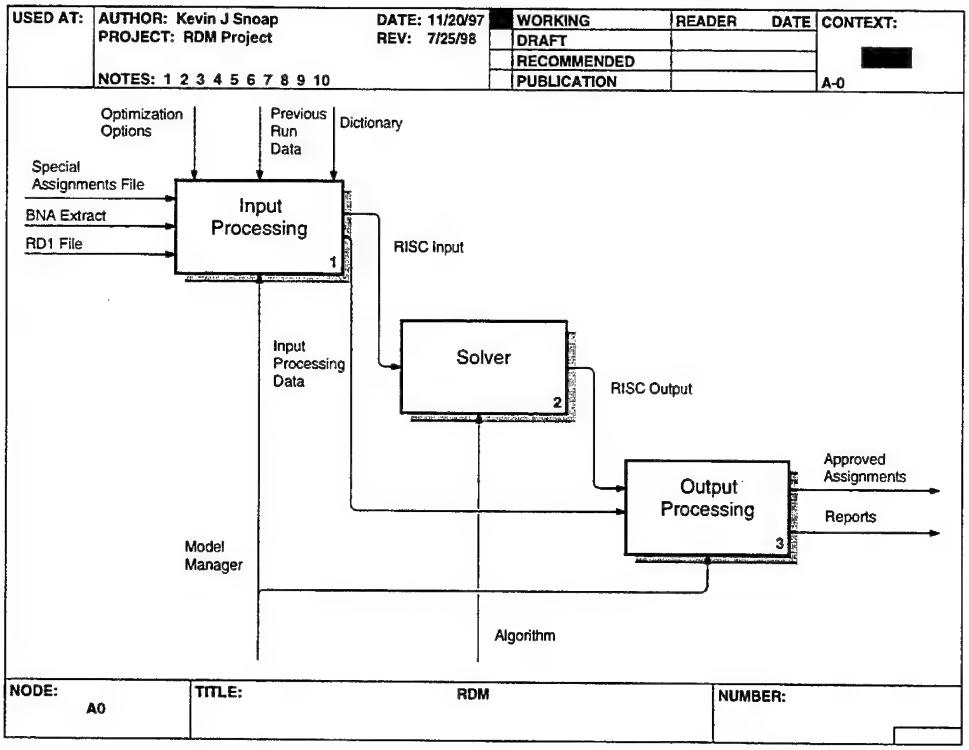
# APPENDIX A. ACRONYMS

**Acronym=>Meaning**

| | |
|---|---|
| AMPL | A Modeling Language for Mathematical Programming |
| ARMS | Automated Recruit Management System |
| ASVAB | Armed Services Vocational Aptitude Battery |
| BNA | By Name Assignment System |
| BPWin | Business Process for Windows |
| COTS | Commercial-off-the-shelf |
| CPlex | Optimization Package for Complex Linear, Network and Integer Programming |
| DSAI | Decision Support Associates, Inc. |
| DSS | Decision Support System |
| ELS | Entry Level School |
| ICOM | Input, Control, Output, Mechanism |
| IDEF | Defense Institute Modeling |
| IDEF0 | Business Process Modeling |
| MCRD | Marine Corps Recruit Depot |
| MCT | Marine Combat Training |
| MODS | Military Operational Data Store |
| MOS | Military Occupational Specialty |
| PEF | Program Enlisted For |
| RD1 | Recruit Distribution Input File |
| RD3 | Recruit Distribution Output File |
| RDdss | Recruit Distribution Decision Support System |
| RDM | Recruit Distribution Model |
| USMC | United States Marine Corps |
| VBA | Visual Basic for Applications |

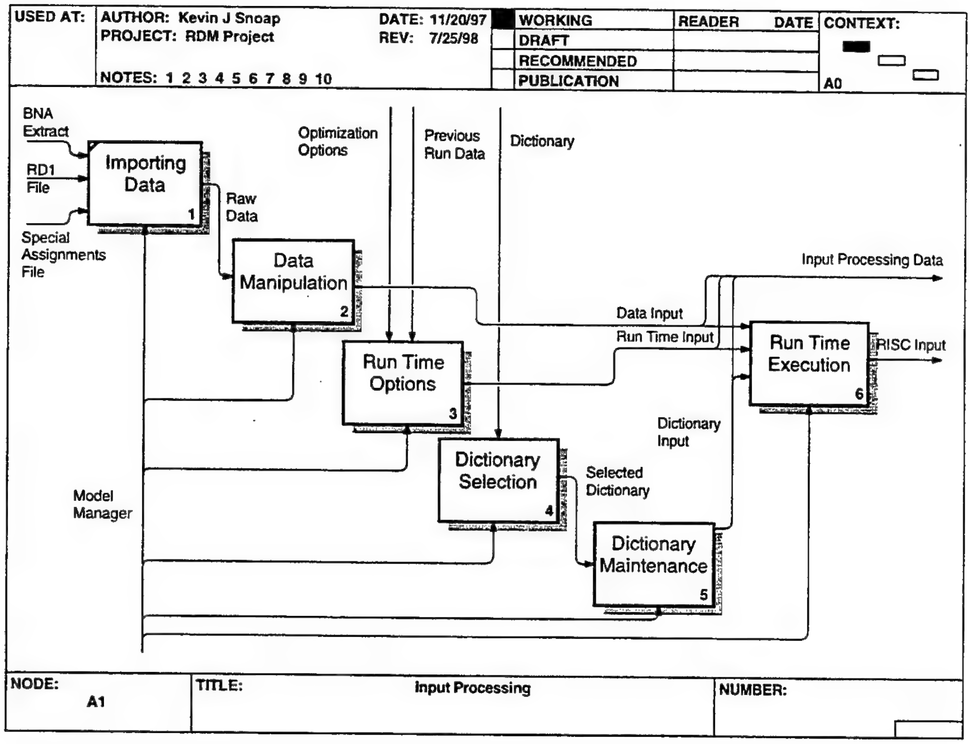
# APPENDIX B. AS-IS RD BUSINESS PROCESS (IDEF0) MODEL





USED AT: AUTHOR: Kevin J Snoap DATE: 11/20/97 ■ WORKING READER DATE CONTEXT:
PROJECT: RDM Project REV: 7/25/98 DRAFT
RECOMMENDED
NOTES: 1 2 3 4 5 6 7 8 9 10 PUBLICATION A-0

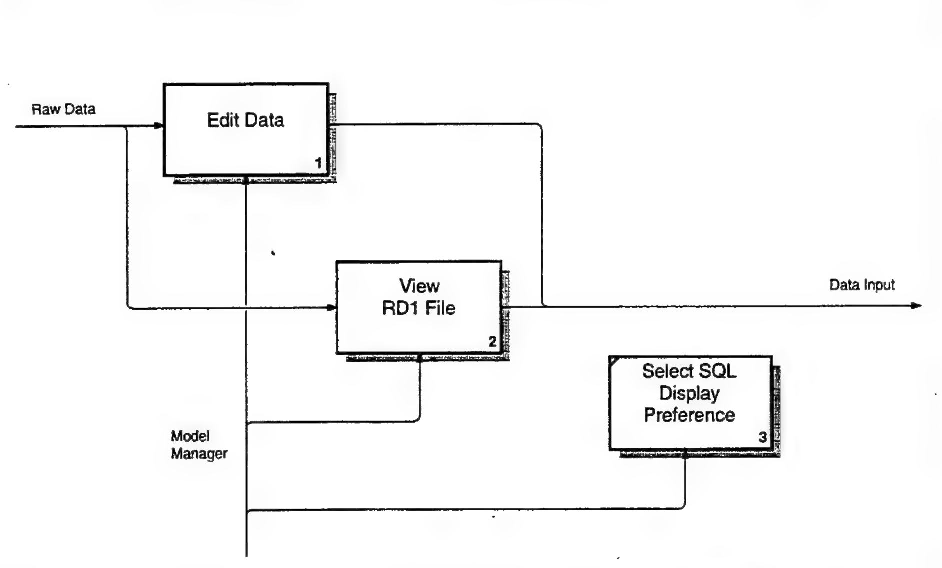
NODE: A0 TITLE: RDM NUMBER:



USED AT: AUTHOR: Kevin J Snoap DATE: 11/20/97 ■ WORKING READER DATE CONTEXT:
PROJECT: RDM Project REV: 7/25/98 DRAFT
RECOMMENDED
NOTES: 1 2 3 4 5 6 7 8 9 10 PUBLICATION A0

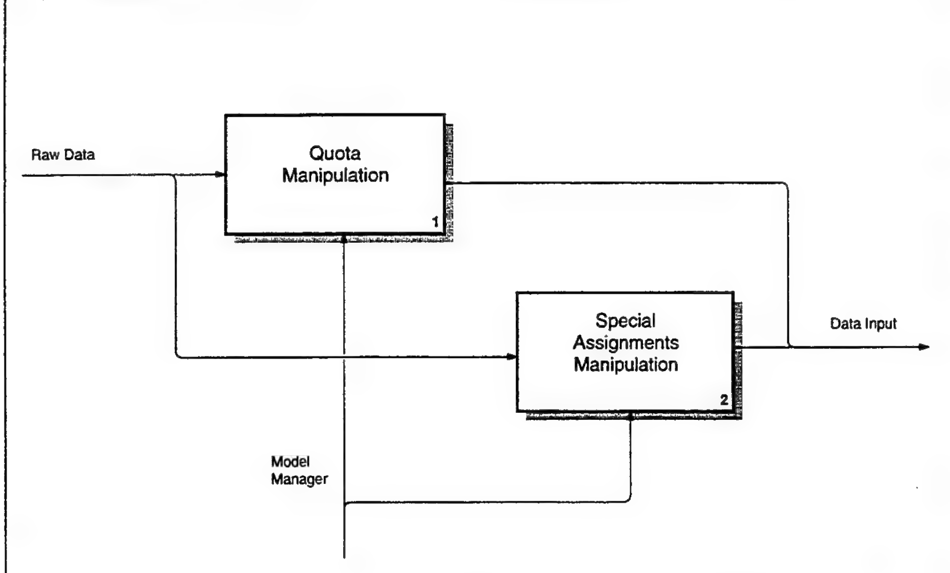
NODE: A1 TITLE: Input Processing NUMBER:

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>□ DRAFT<br>□ RECOMMENDED<br>□ PUBLICATION | READER DATE | CONTEXT:<br>A1 |



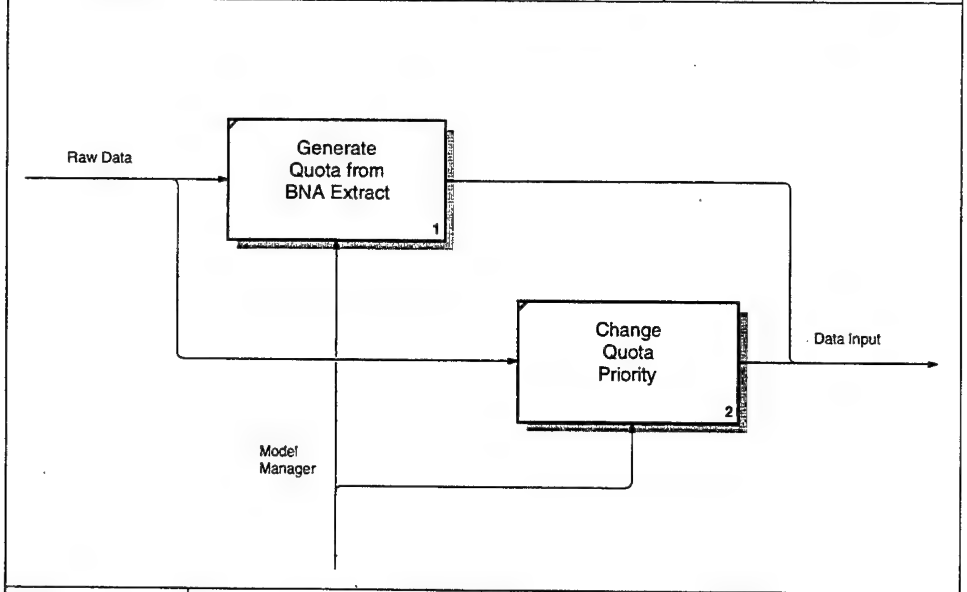
| | | |
|---|---|---|
| NODE: A1.2 | TITLE: Data Manipulation | NUMBER: |

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>□ DRAFT<br>□ RECOMMENDED<br>□ PUBLICATION | READER DATE | CONTEXT:<br>A1.2 |



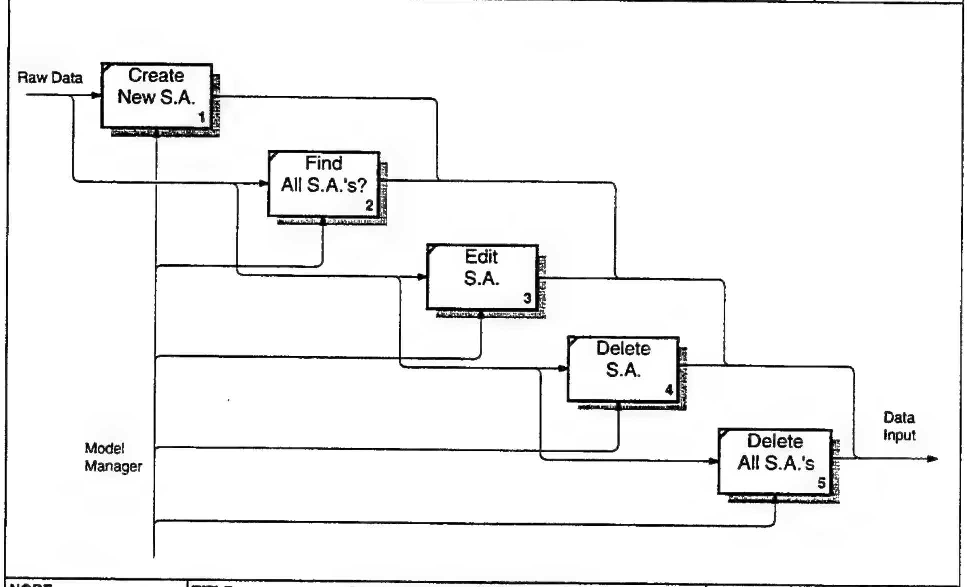
| | | |
|---|---|---|
| NODE: A1.2.1 | TITLE: Edit Data | NUMBER: |

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>DRAFT<br>RECOMMENDED<br>PUBLICATION | READER DATE | CONTEXT:<br>A1.2.1 |



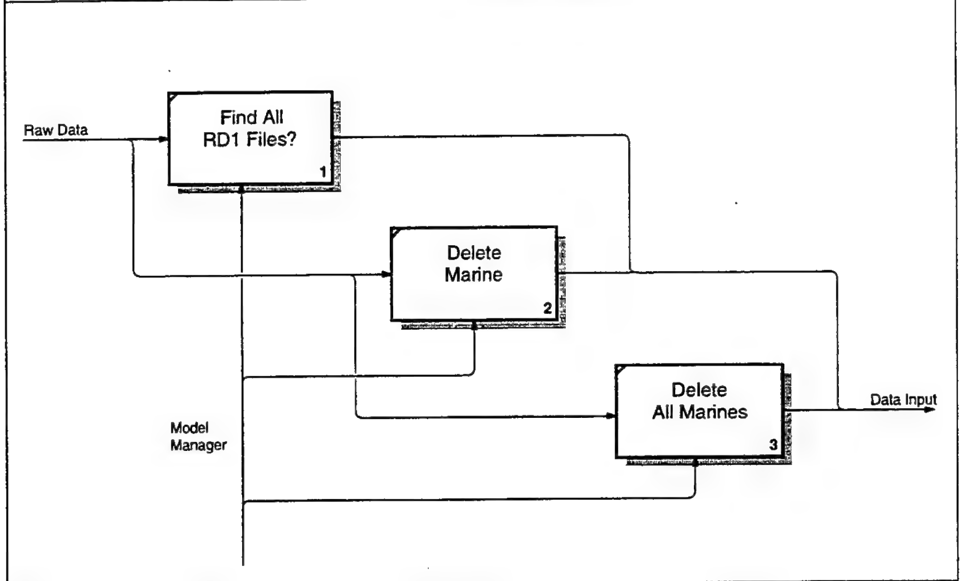
| | | |
|---|---|---|
| NODE: A1.2.1.1 | TITLE: Quota Manipulation | NUMBER: |

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>DRAFT<br>RECOMMENDED<br>PUBLICATION | READER DATE | CONTEXT:<br>A1.2.1 |



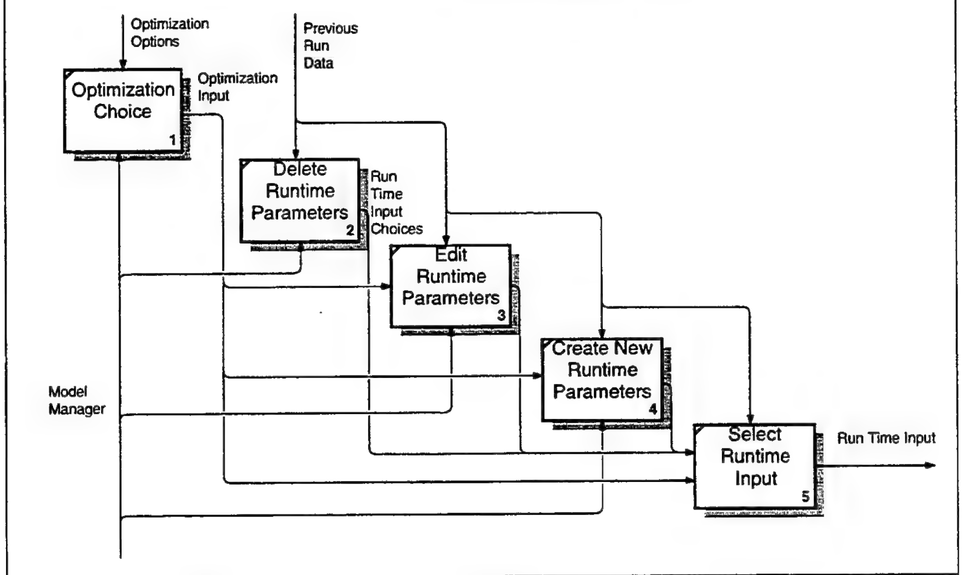
| | | |
|---|---|---|
| NODE: A1.2.1.2 | TITLE: Special Assignments Manipulation | NUMBER: |

| | | | | | | |
|---|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>DRAFT<br>RECOMMENDED<br>PUBLICATION | READER | DATE | CONTEXT:<br>A1.2 |

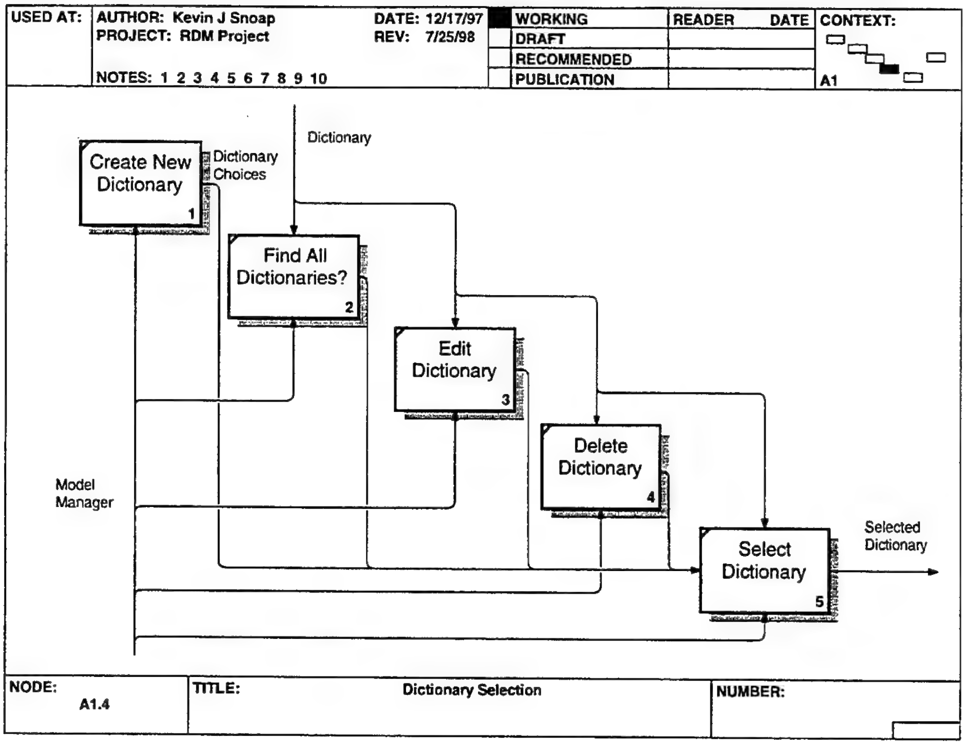


| | | |
|---|---|---|
| NODE: A1.2.2 | TITLE: View RD1 File | NUMBER: |

| | | | | | | |
|---|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/10/97<br>REV: 7/25/98 | ■ WORKING<br>DRAFT<br>RECOMMENDED<br>PUBLICATION | READER | DATE | CONTEXT:<br>A1 |



| | | |
|---|---|---|
| NODE: A1.3 | TITLE: Run Time Options | NUMBER: |



USED AT:

AUTHOR: Kevin J Snoap

PROJECT: RDM Project

DATE: 12/17/97

REV: 7/25/98

NOTES: 1 2 3 4 5 6 7 8 9 10

■ WORKING

DRAFT

RECOMMENDED

PUBLICATION

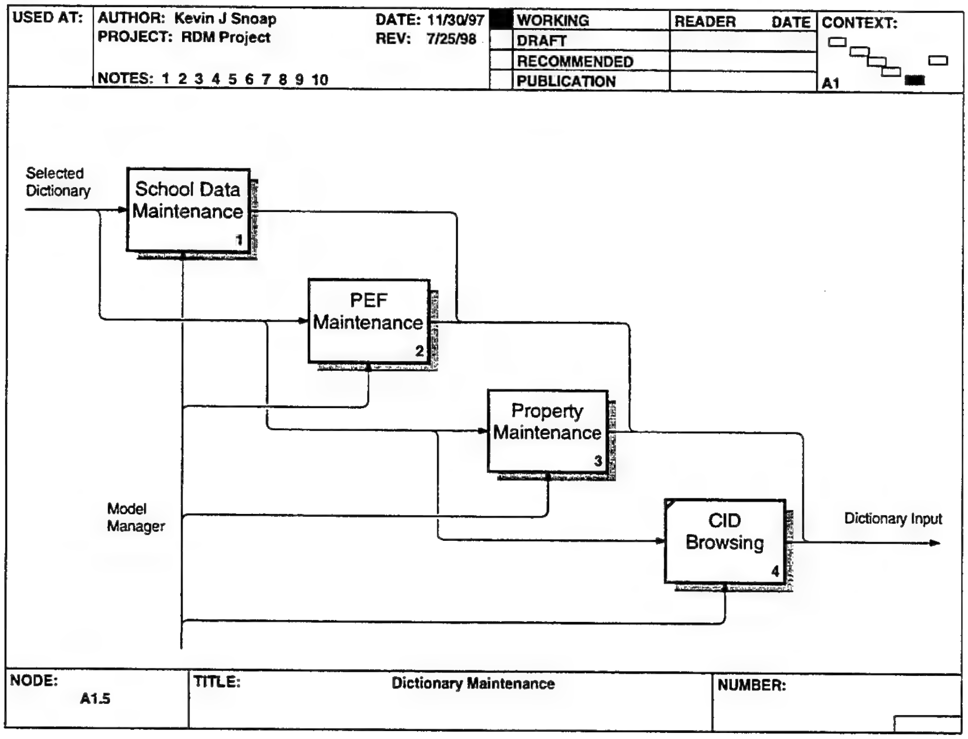
READER DATE

CONTEXT: A1

NODE: A1.4

TITLE: Dictionary Selection

NUMBER:



USED AT:

AUTHOR: Kevin J Snoap

PROJECT: RDM Project

DATE: 11/30/97

REV: 7/25/98

NOTES: 1 2 3 4 5 6 7 8 9 10

■ WORKING

DRAFT

RECOMMENDED

PUBLICATION

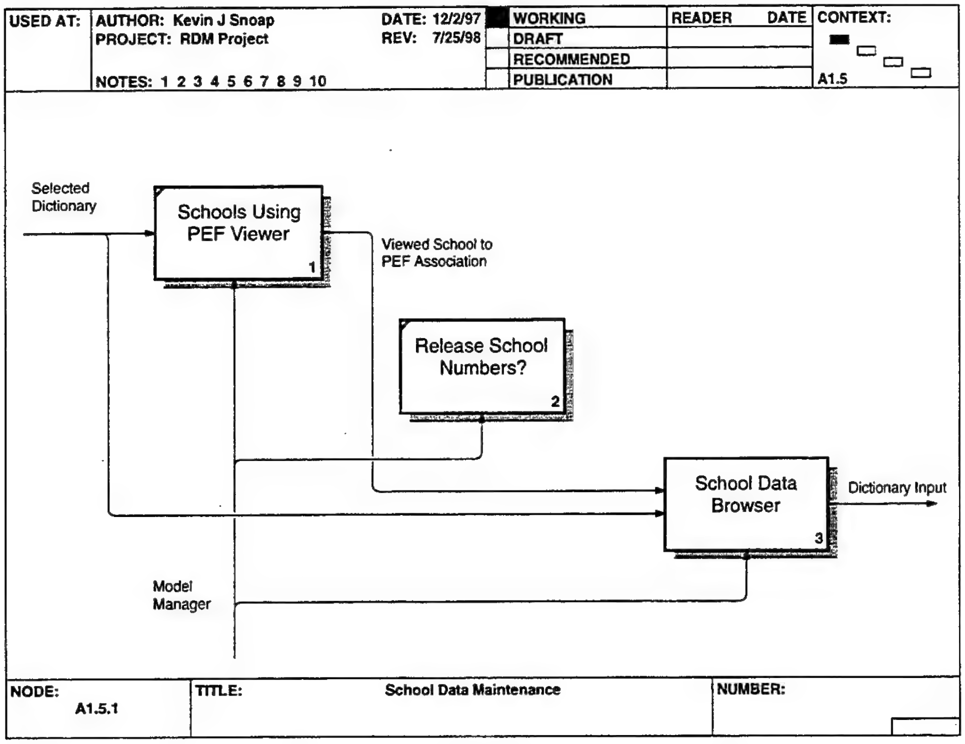
READER DATE

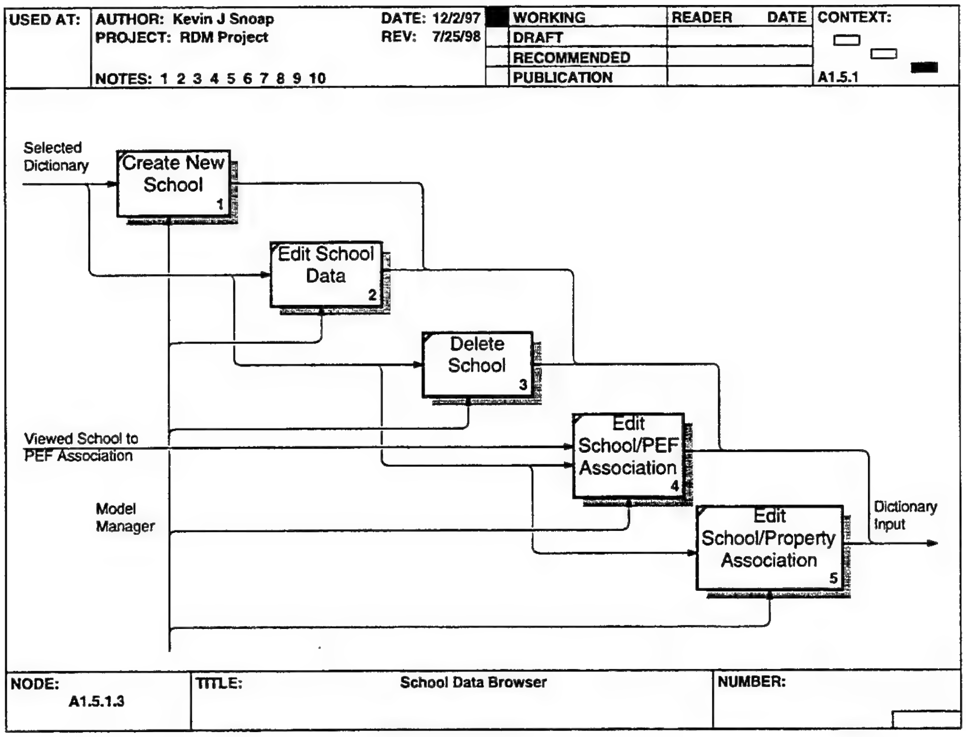
CONTEXT: A1

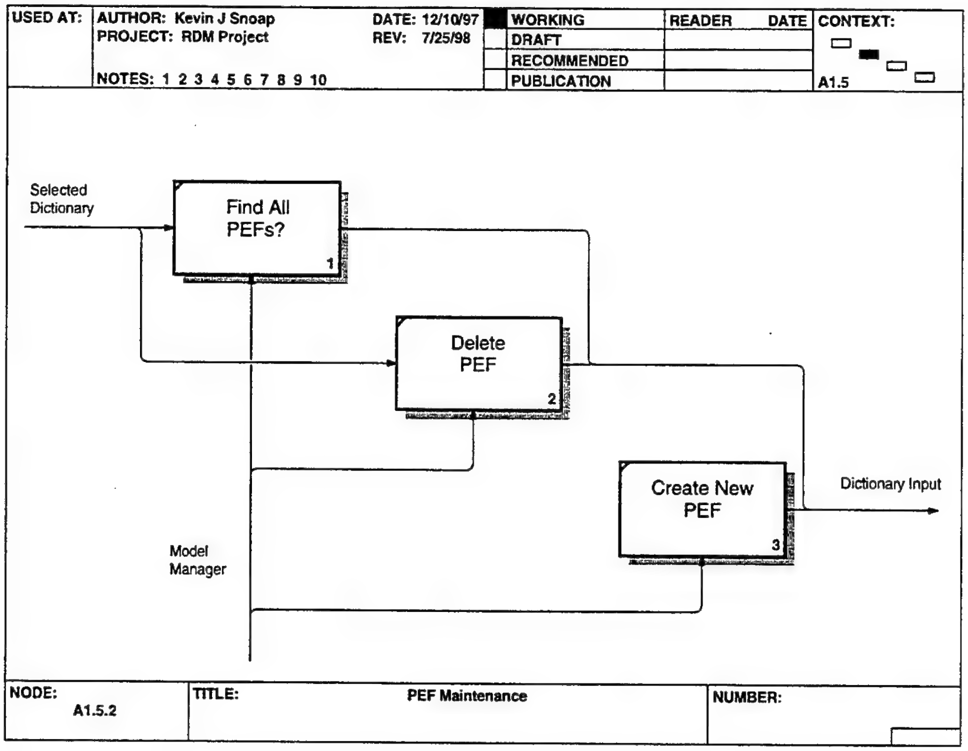
NODE: A1.5

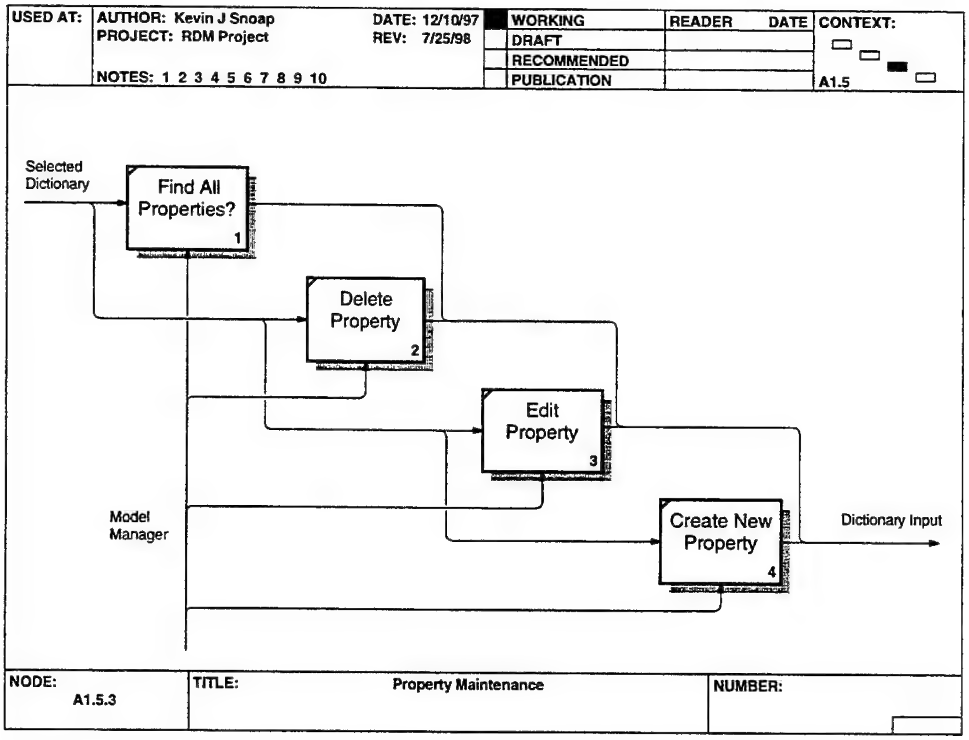
TITLE: Dictionary Maintenance

NUMBER:

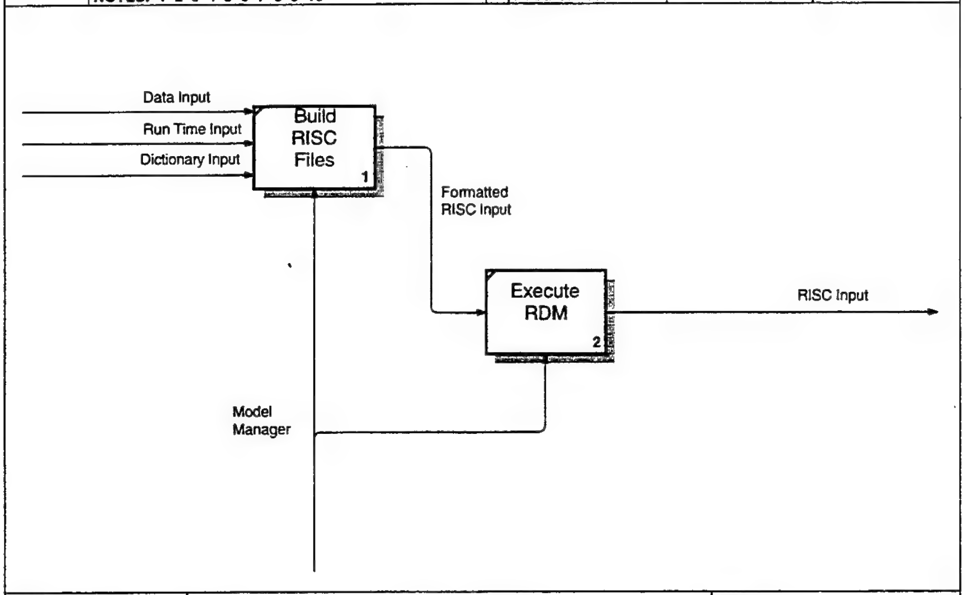

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 12/2/97
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A1.5
Selected Dictionary
Schools Using PEF Viewer
1
Viewed School to PEF Association
Release School Numbers?
2
School Data Browser
3
Dictionary Input
Model Manager
NODE:
A1.5.1
TITLE:
School Data Maintenance
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 12/2/97
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A1.5.1
Selected Dictionary
Create New School
1
Edit School Data
2
Delete School
3
Viewed School to PEF Association
Edit School/PEF Association
4
Model Manager
Edit School/Property Association
5
Dictionary Input
NODE:
A1.5.1.3
TITLE:
School Data Browser
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 12/10/97
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A1.5
Selected Dictionary
Find All PEFs?
1
Delete PEF
2
Create New PEF
3
Dictionary Input
Model Manager
NODE:
A1.5.2
TITLE:
PEF Maintenance
NUMBER:

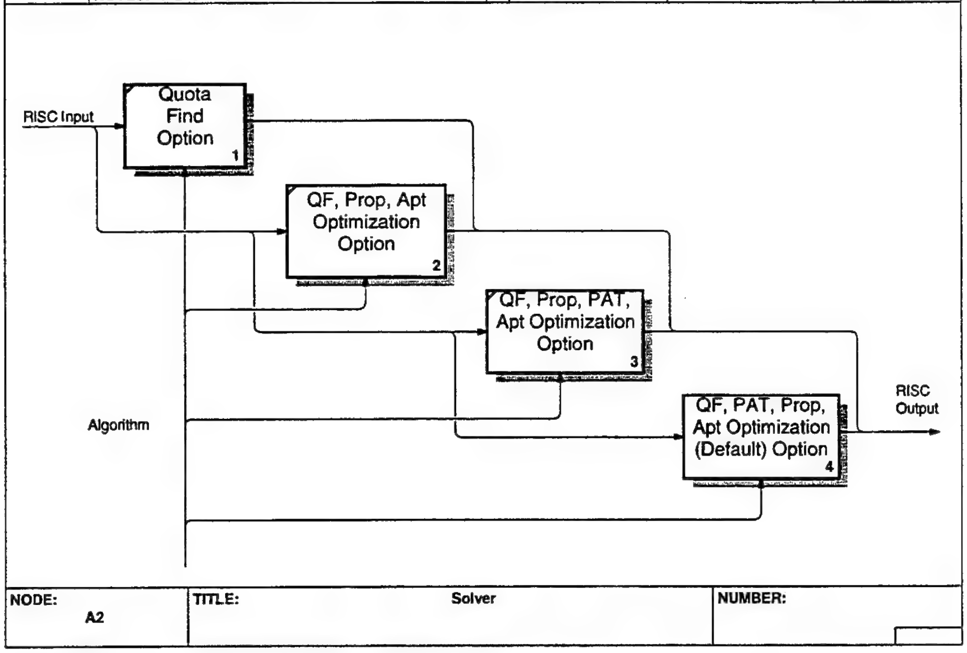

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RDM Project
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 12/10/97
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A1.5
Selected Dictionary
Find All Properties?
1
Delete Property
2
Edit Property
3
Create New Property
4
Dictionary Input
Model Manager
NODE:
A1.5.3
TITLE:
Property Maintenance
NUMBER:

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/10/97<br>REV: 7/25/98 | ■ WORKING<br>□ DRAFT<br>□ RECOMMENDED<br>□ PUBLICATION | READER DATE | CONTEXT:<br>A1 |



| | | |
|---|---|---|
| NODE: A1.6 | TITLE: Run Time Execution | NUMBER: |

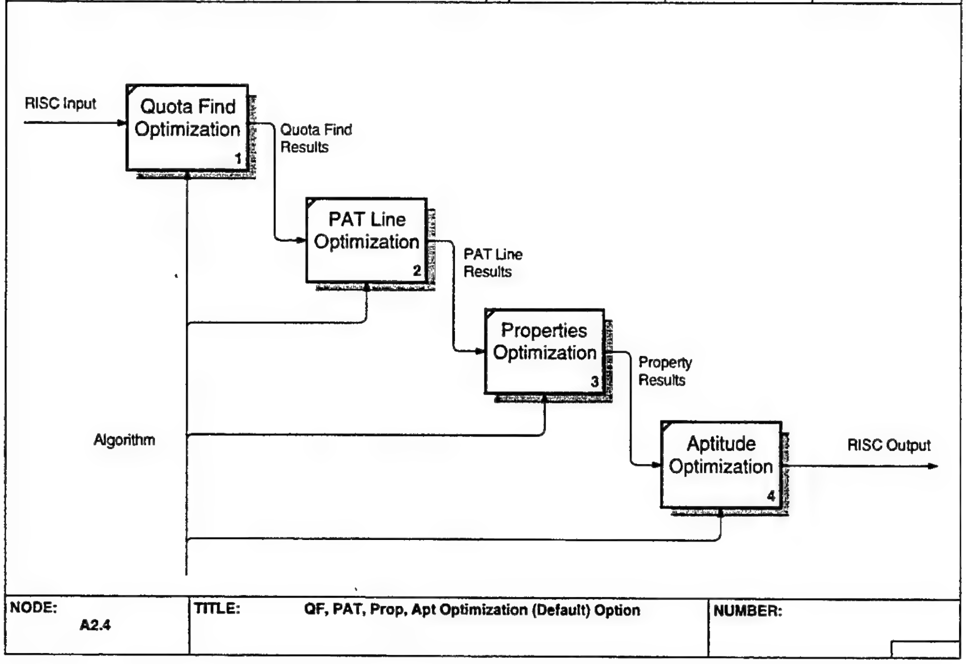
| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/15/97<br>REV: 7/25/98 | ■ WORKING<br>□ DRAFT<br>□ RECOMMENDED<br>□ PUBLICATION | READER DATE | CONTEXT:<br>A0 |



RISC Input
Quota Find Option
1
QF, Prop, Apt Optimization Option
2
QF, Prop, PAT, Apt Optimization Option
3
QF, PAT, Prop, Apt Optimization (Default) Option
4
RISC Output
Algorithm

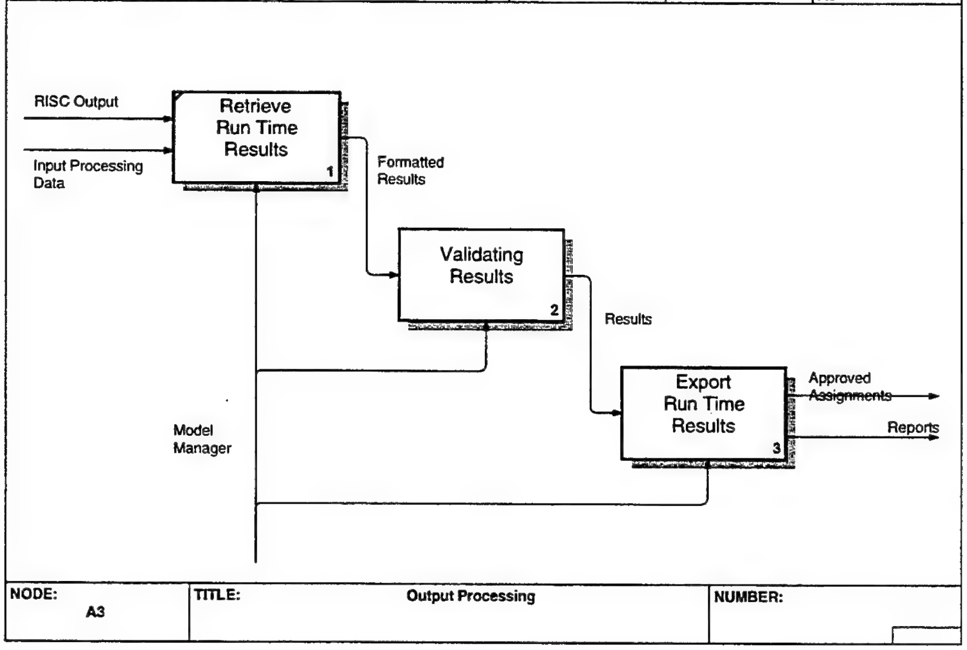
| | | |
|---|---|---|
| NODE: A2 | TITLE: Solver | NUMBER: |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project | DATE: 12/16/97<br>REV: 7/25/98 | ■ WORKING | READER | DATE | CONTEXT: |
|  |  |  | DRAFT |  |  |  |
|  |  |  | RECOMMENDED |  |  |  |
|  | NOTES: 1 2 3 4 5 6 7 8 9 10 |  | PUBLICATION |  |  | A2 |



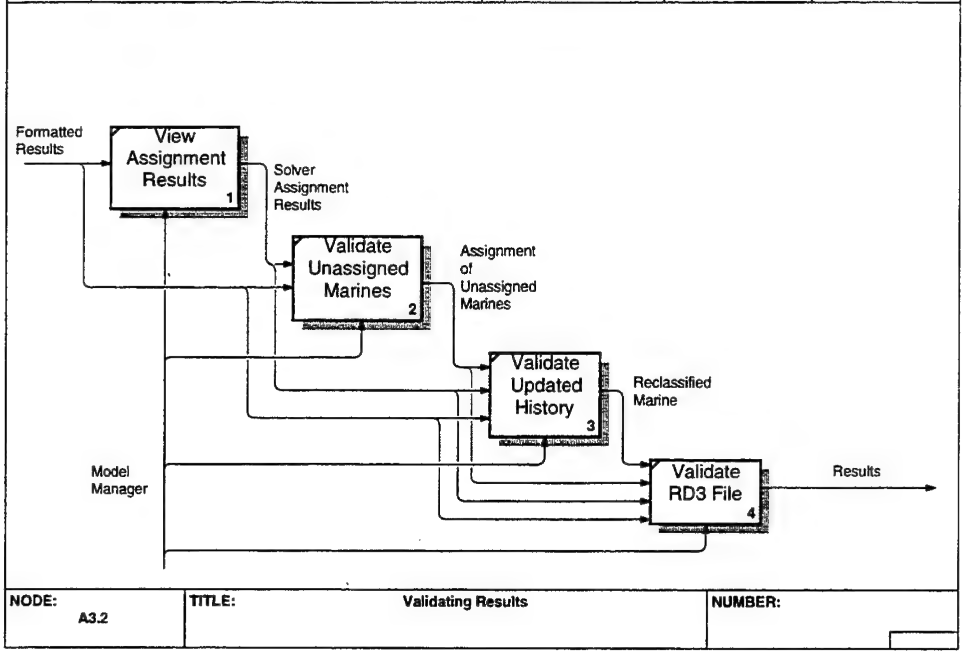
|  |  |  |
|---|---|---|
| NODE: A2.4 | TITLE: QF, PAT, Prop, Apt Optimization (Default) Option | NUMBER: |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING | READER | DATE | CONTEXT: |
|  |  |  | DRAFT |  |  |  |
|  |  |  | RECOMMENDED |  |  |  |
|  | NOTES: 1 2 3 4 5 6 7 8 9 10 |  | PUBLICATION |  |  | A0 |



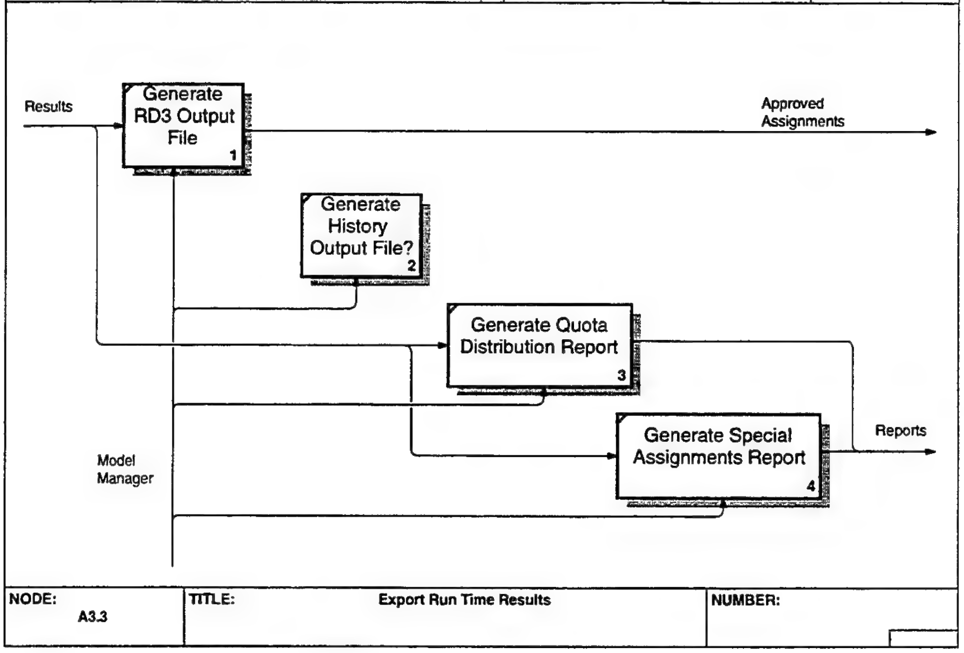
|  |  |  |
|---|---|---|
| NODE: A3 | TITLE: Output Processing | NUMBER: |

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>DRAFT<br>RECOMMENDED<br>PUBLICATION | READER DATE | CONTEXT:<br>A3 |



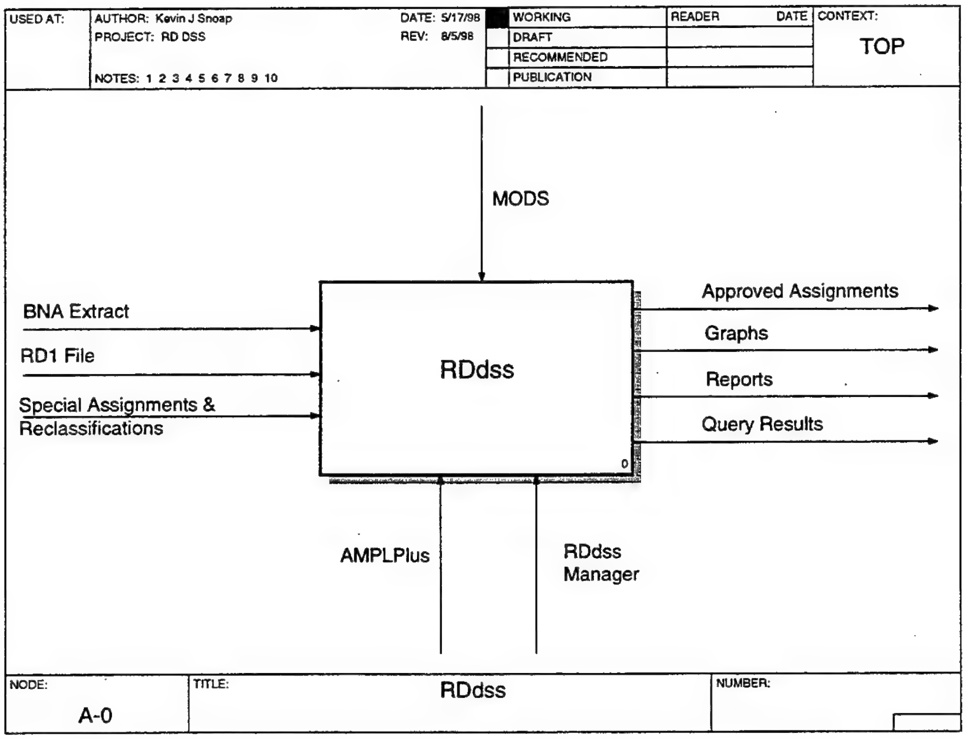
| | | |
|---|---|---|
| NODE: A3.2 | TITLE: Validating Results | NUMBER: |

| | | | | | |
|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RDM Project<br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 12/11/97<br>REV: 7/25/98 | ■ WORKING<br>DRAFT<br>RECOMMENDED<br>PUBLICATION | READER DATE | CONTEXT:<br>A3 |


Results
Generate RD3 Output File
1
Approved Assignments
Generate History Output File?
2
Generate Quota Distribution Report
3
Generate Special Assignments Report
4
Reports
Model Manager

| | | |
|---|---|---|
| NODE: A3.3 | TITLE: Export Run Time Results | NUMBER: |

# APPENDIX C. TO-BE RD BUSINESS PROCESS (IDEF0) MODEL



USED AT:

AUTHOR: Kevin J Snoap DATE: 5/17/98
PROJECT: RD DSS REV: 8/5/98

NOTES: 1 2 3 4 5 6 7 8 9 10

■ WORKING
□ DRAFT
□ RECOMMENDED
□ PUBLICATION

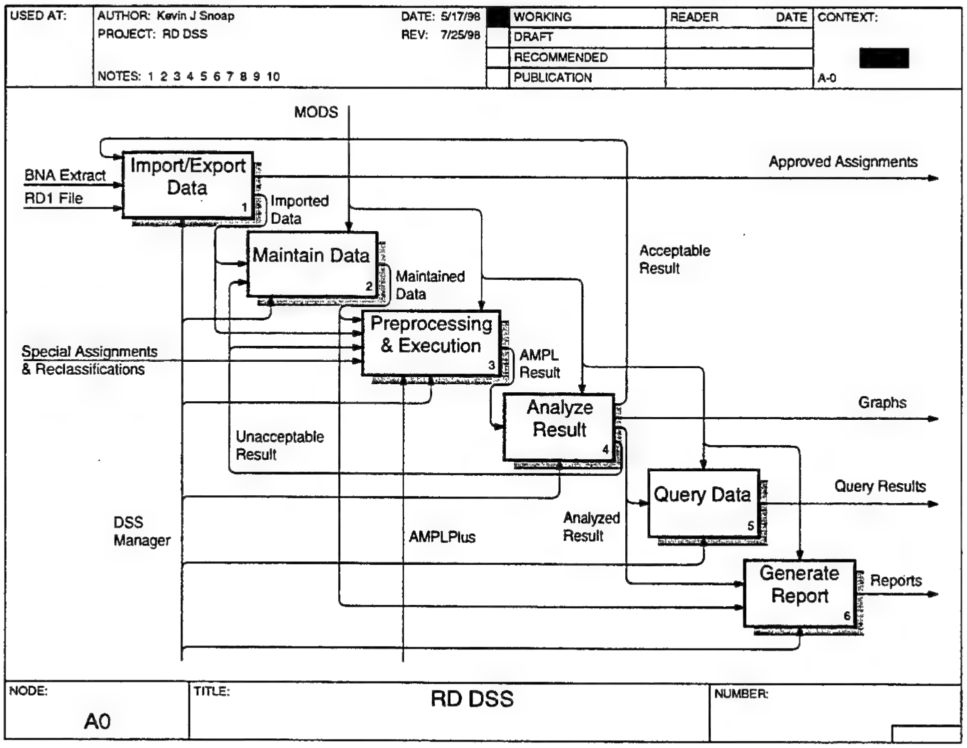
READER DATE

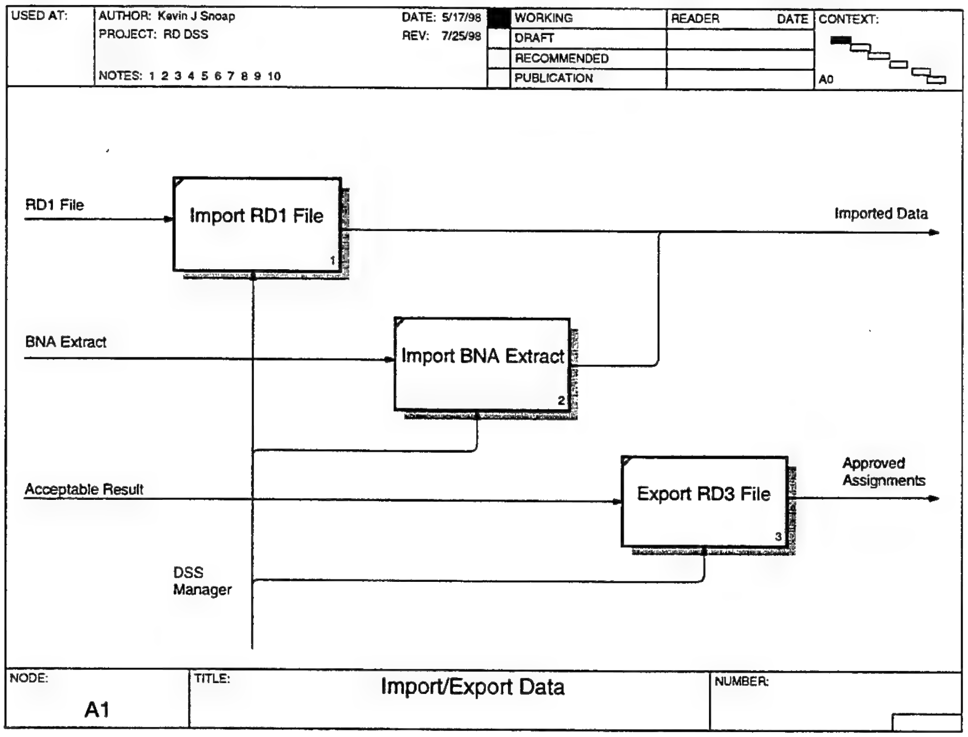
CONTEXT: TOP

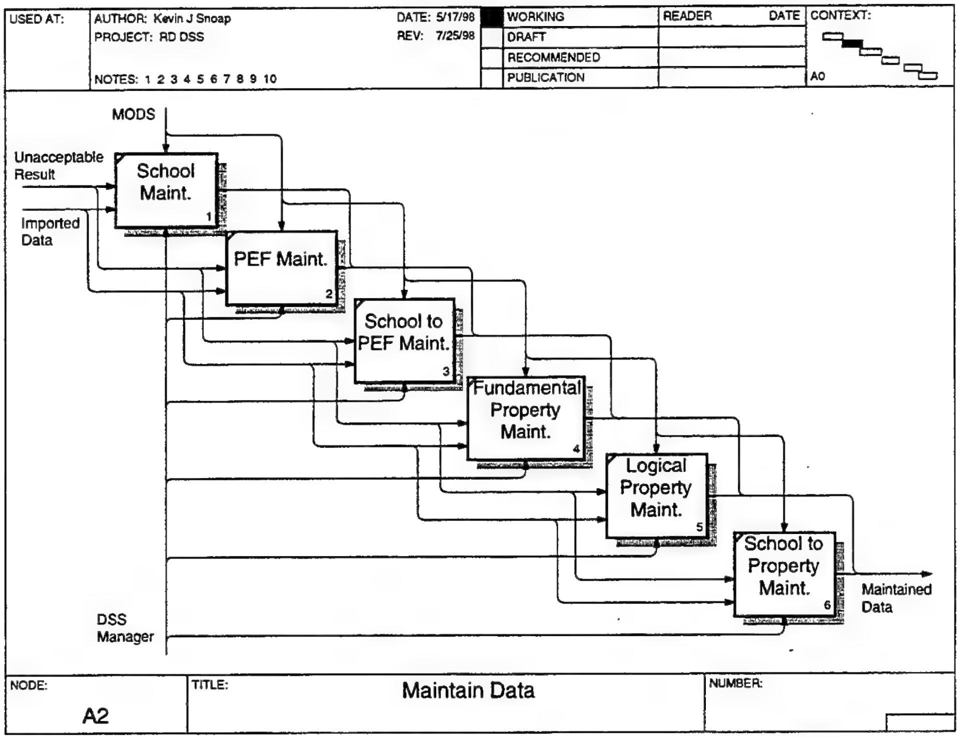
NODE: A-0

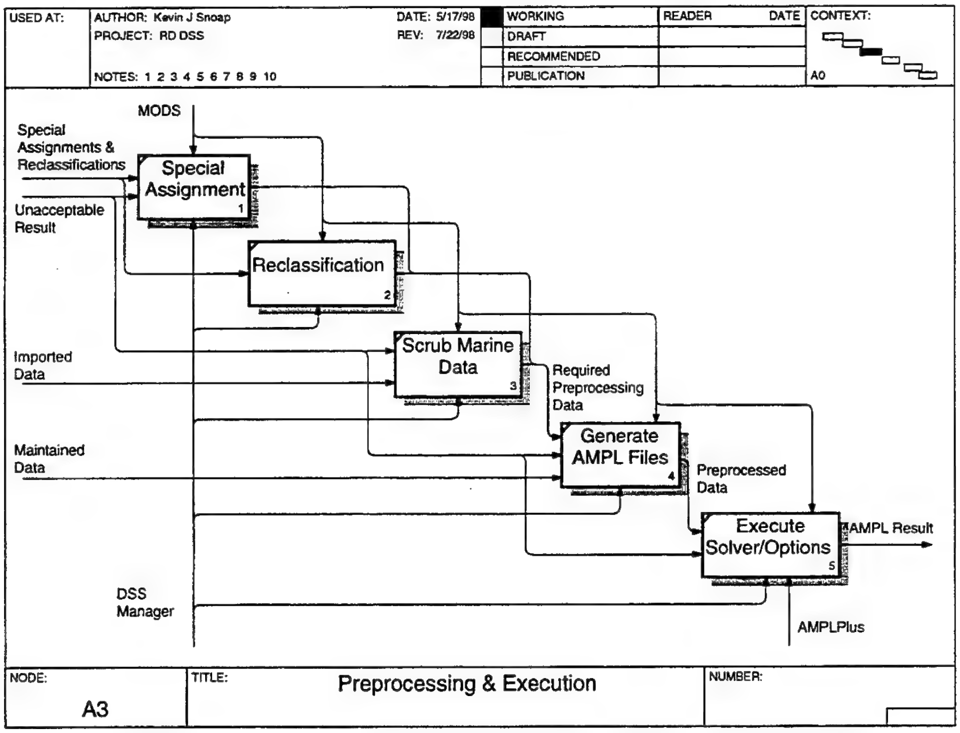
TITLE: RDdss

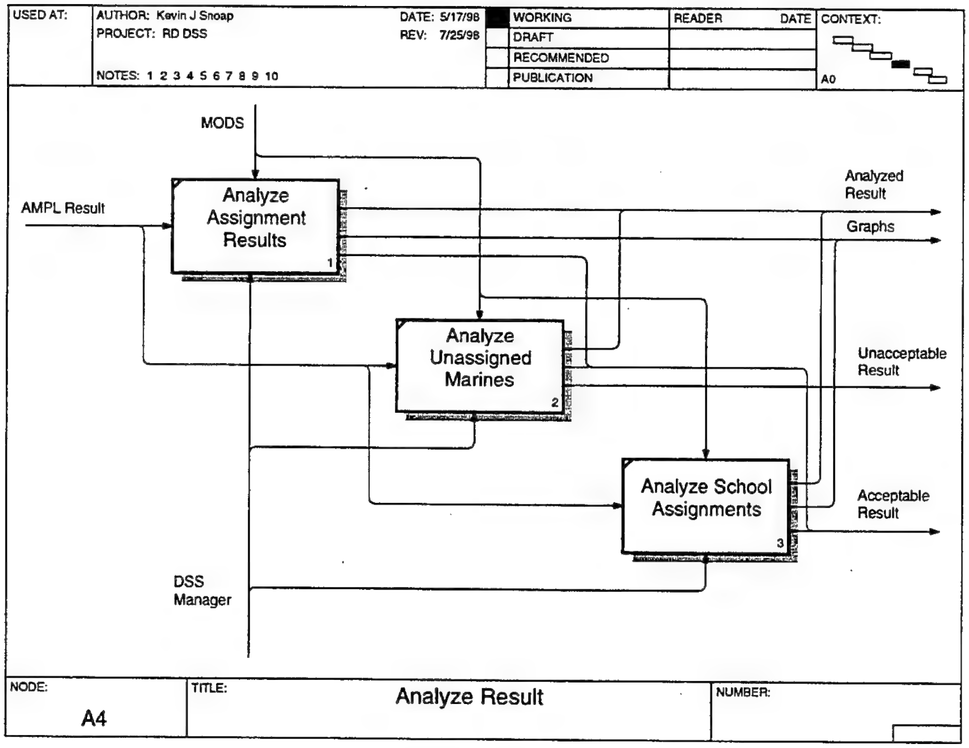
NUMBER:

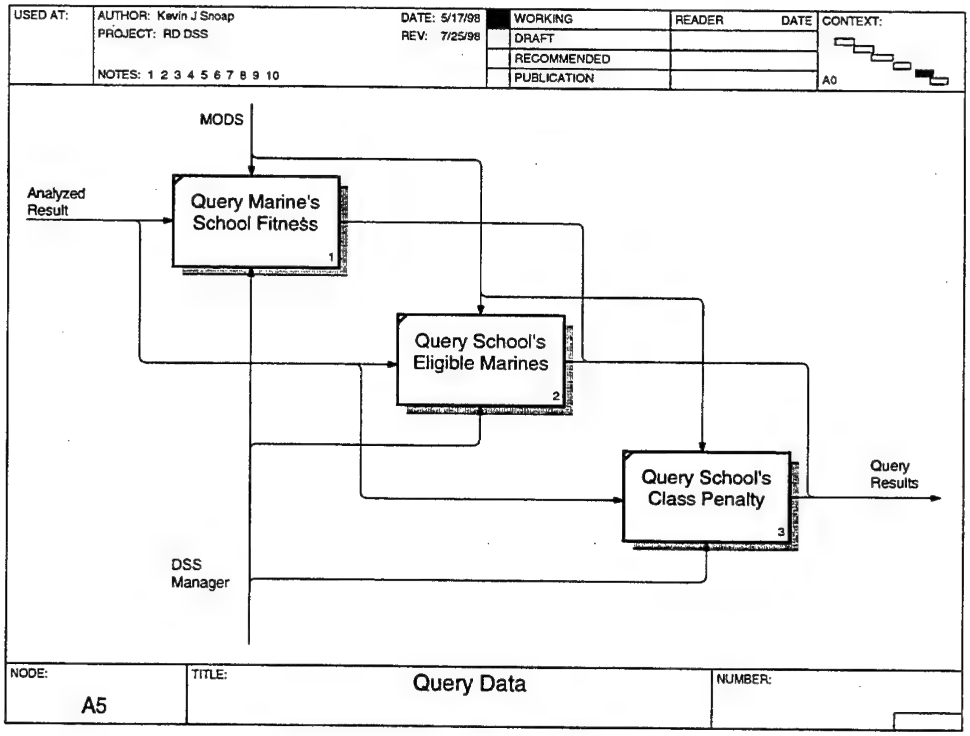

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A-0
MODS
BNA Extract
RD1 File
Import/Export Data
1
Imported Data
Approved Assignments
Maintain Data
2
Maintained Data
Acceptable Result
Special Assignments & Reclassifications
Preprocessing & Execution
3
AMPL Result
Analyze Result
4
Graphs
Unacceptable Result
Query Data
5
Query Results
DSS Manager
AMPLPlus
Analyzed Result
Generate Report
6
Reports
NODE:
A0
TITLE:
RD DSS
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
RD1 File
Import RD1 File
1
Imported Data
BNA Extract
Import BNA Extract
2
Acceptable Result
Export RD3 File
3
Approved Assignments
DSS Manager
NODE:
A1
TITLE:
Import/Export Data
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Unacceptable Result
Imported Data
School Maint.
1
PEF Maint.
2
School to PEF Maint.
3
Fundamental Property Maint.
4
Logical Property Maint.
5
School to Property Maint.
6
Maintained Data
DSS Manager
NODE:
A2
TITLE:
Maintain Data
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/22/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Special Assignments & Reclassifications
Unacceptable Result
Special Assignment
1
Reclassification
2
Imported Data
Scrub Marine Data
3
Required Preprocessing Data
Maintained Data
Generate AMPL Files
4
Preprocessed Data
Execute Solver/Options
5
AMPL Result
DSS Manager
AMPLPlus
NODE:
A3
TITLE:
Preprocessing & Execution
NUMBER:

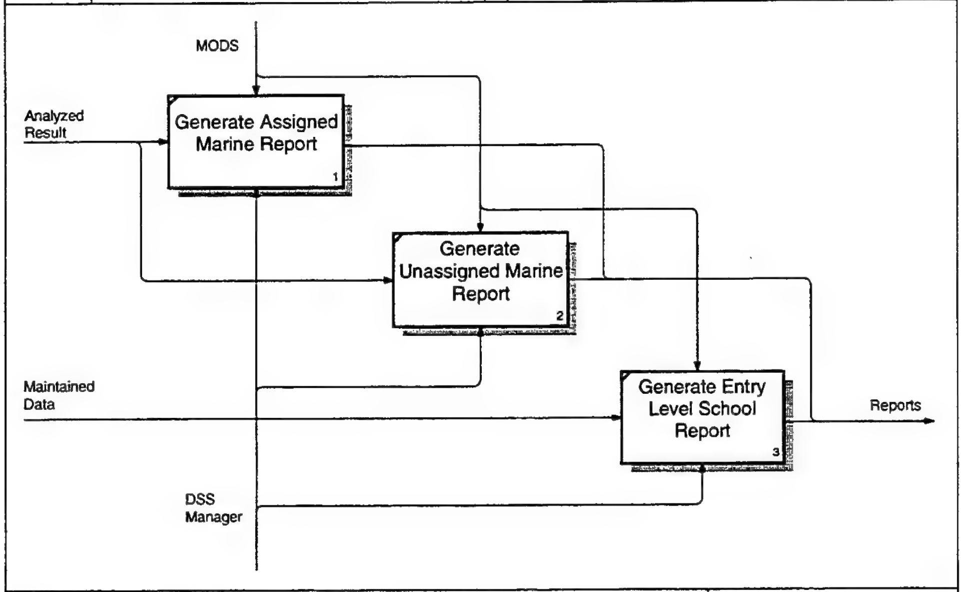

USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
AMPL Result
Analyze Assignment Results
1
Analyze Unassigned Marines
2
Analyze School Assignments
3
Analyzed Result
Graphs
Unacceptable Result
Acceptable Result
DSS Manager
NODE:
A4
TITLE:
Analyze Result
NUMBER:


USED AT:
AUTHOR: Kevin J Snoap
PROJECT: RD DSS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 5/17/98
REV: 7/25/98
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
MODS
Analyzed Result
Query Marine's School Fitness
1
Query School's Eligible Marines
2
Query School's Class Penalty
3
Query Results
DSS Manager
NODE:
A5
TITLE:
Query Data
NUMBER:

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| USED AT: | AUTHOR: Kevin J Snoap<br>PROJECT: RD DSS<br><br>NOTES: 1 2 3 4 5 6 7 8 9 10 | DATE: 5/17/98<br>REV: 7/22/98 | ■ | WORKING | READER | DATE |
|  |  |  |  | DRAFT |  |  |
|  |  |  |  | RECOMMENDED |  |  |
|  |  |  |  | PUBLICATION |  |  |

CONTEXT:

A0


MODS
Analyzed Result
Generate Assigned Marine Report
1
Generate Unassigned Marine Report
2
Maintained Data
Generate Entry Level School Report
3
Reports
DSS Manager

NODE: A6

TITLE: Generate Report

NUMBER:

# APPENDIX D. RDDSS VBA CODE

```
Form: frmAnalyzeResult
    Code
\Attribute VB_Name = "Form_frmAnalyzeResult"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Dim booSchoolAssignment As Boolean


Private Sub btnExit_Click()
On Error GoTo Err_btnExit_Click


    DoCmd.Close

Exit_btnExit_Click:
    Exit Sub

Err_btnExit_Click:
    MsgBox Err.Description
    Resume Exit_btnExit_Click


End Sub

Private Sub btnSchoolAssignments_Click()
On Error GoTo Err_btnSchoolAssignments_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Set the flag
    booSchoolAssignment = True

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmSchoolAssignments"
    DoCmd.OpenForm stDocName, , , stLinkCriteria


Exit_btnSchoolAssignments_Click:
    Exit Sub

Err_btnSchoolAssignments_Click:
    MsgBox Err.Description
    Resume Exit_btnSchoolAssignments_Click

End Sub

Private Sub btnSolverAssignments_Click()
On Error GoTo Err_btnSolverAssignments_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmAssignmentResult"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnSolverAssignments_Click:
    Exit Sub

Err_btnSolverAssignments_Click:
    MsgBox Err.Description
    Resume Exit_btnSolverAssignments_Click

End Sub

Private Sub btnUnassigedMarines_Click()
On Error GoTo Err_btnUnassigedMarines_Click


    Dim strSQL As String
    Dim rec As Recordset
    Dim db1 As Database

    Set db1 = CurrentDb()

    ' If the number of unassigned marines = 0, then disable the
unassigned marines button
    strSQL = "SELECT Count(PEF) AS TotalUnassigned FROM
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    If rec!TotalUnassigned = 0 Then
        MsgBox ("                    All Marines were assigned.")
    Else

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmUnassignedMarines"
DoCmd.OpenForm stDocName, , , stLinkCriteria

    End If


Exit_btnUnassigedMarines_Click:
    Exit Sub

Err_btnUnassigedMarines_Click:
    MsgBox Err.Description
    Resume Exit_btnUnassigedMarines_Click

End Sub

Private Sub Command8_Click()
On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmRDM_Main_Switchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

Err_Command8_Click:
    MsgBox Err.Description
    Resume Exit_Command8_Click

End Sub


Private Sub Form_Deactivate()

    If booSchoolAssignment Then

        Dim strSQL As String
        Dim i As Integer, j As Integer
        Dim rec As Recordset, rec1 As Recordset, rec2 As Recordset
        Dim db1 As Database

        Set db1 = CurrentDb()

        ' Create a table to get the fit and fill weights used during the
        Application.SetOption "Confirm Action Queries", False
        ' Delete the table if it already exists
        DeleteTable

        db1.Execute "CREATE TABLE PERCENTAGE (CourseNumber TEXT, AMOS
TEXT, Quota INTEGER, Assigned INTEGER, Percentage INTEGER);"
        Application.SetOption "Confirm Action Queries", True

        strSQL = "SELECT * FROM PERCENTAGE;"
        Set rec = db1.OpenRecordset(strSQL, dbOpenDynaset)

        strSQL = "SELECT * FROM qryTotalQuotaForRun ORDER BY
        Set rec1 = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        strSQL = "SELECT * FROM qryTotalQuotaFilledForRun ORDER BY
SCourseNumber_FK;"
        Set rec2 = db1.OpenRecordset(strSQL, dbOpenSnapshot)
        ' Ensure there are values in the records.
        If rec1.EOF = False Then
            If rec2.EOF = False Then
                rec1.MoveLast
                rec1.MoveFirst
                rec2.MoveLast
                rec2.MoveFirst
                ' Fill the PENALTY table with the quota values for the
                For i = 1 To rec1.RecordCount
                    rec.AddNew
                    rec!CourseNumber = rec1!SCourseNumber_FK
                    rec!AMOS = rec1!AMOS_FK
                    rec!Quota = rec1!SumOFQuota
                    rec.Update
                    rec1.MoveNext
                Next i
                rec.MoveFirst
                ' Update the PENALTY table with the percentage of quota
                For i = 1 To rec.RecordCount
                    rec2.MoveFirst
                    For j = 1 To rec2.RecordCount
                        If rec!CourseNumber = rec2!SCourseNumber_FK Then
                            If rec!AMOS = rec2!AMOS_FK Then
                                rec.Edit
                                rec!Assigned = rec2!CountOfFiscalYear_PK
                                rec!Percentage = Int((rec!Assigned /
rec!Quota) * 100)
                                rec.Update
                                Exit For
                            End If
                        End If
                        rec2.MoveNext
                        ' If no match was found, then the percent fill is
                        If j = rec2.RecordCount Then
                            rec.Edit
                            rec!Assigned = 0
                            rec!Percentage = 0
                            rec.Update
                        End If
                    Next j
                    rec.MoveNext
                Next i

            End If

        End If

        rec.Close
        rec1.Close
        rec2.Close
        db1.Close

        ' Reset the flag
        booSchoolAssignment = False
    Else
        ' Delete the PERCENTAGE table
        DeleteTable
    End If

End Sub


Form: frmAssignmentResult
    Code
Attribute VB_Name = "Form_frmAssignmentResult"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit


Private Sub Form_Current()

    Dim strSQL As String
    Dim rec As Recordset
    Dim db1 As Database

    Set db1 = CurrentDb()

    ' Calculates the high, average, and low fitness scores. Also, it
calculates the number of solver assignments.
    strSQL = "SELECT Avg(Fitness) AS Average, Max(Fitness) As Maximum,
Min(Fitness) As Minimum, Count(Fitness) As TotalAssigned FROM
qryAssignmentResult;"
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtFitnessAvg = Int(rec!Average)
    Me.txtFitnessMax = rec!Maximum
    Me.txtFitnessMin = rec!Minimum
    Me.txtSolverAssignments = rec!TotalAssigned

    ' Calculates the number of manual assignments, and the total number
of assignments.
    strSQL = "SELECT Count(AssignmentType) AS TotalManualAssigned FROM
qryManuallyAssignedMarines;"
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtManualAssignments = rec!TotalManualAssigned
    Me.txtSolverAssignments

    ' Calculates the number of unassigned marines, and the total number
of assignable marines.
    strSQL = "SELECT Count(PEF) AS TotalUnassigned FROM
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtTotalUnassigned = rec!TotalUnassigned
    Me.txtTotalAssignable = rec!TotalUnassigned +

    ' Create a table to get the fit and fill weights used during the run
    Application.SetOption "Confirm Action Queries", False
    db1.Execute "CREATE TABLE CONSTANTS (ConstantName TEXT, intValue
    DoCmd.TransferText acImportFixed, "Rdm_Data Export Specification",
CONSTANTS, "C:\ToBeRdm\AmplData\rdm.dat"
    Application.SetOption "Confirm Action Queries", True

    strSQL = "SELECT IntValue FROM CONSTANTS;"
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtFitWeight = rec!IntValue
    rec.MoveNext
    rec.MoveNext
    Me.txtFillWeight = rec!IntValue
    rec.Close

    ' Remove the created table
    db1.Execute "DROP TABLE CONSTANTS;"
    db1.Close


End Sub
Private Sub Close_Click()
On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmAnalyzeResult"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub

Form: frmChangeGradDate
    Code
Attribute VB_Name = "Form_frmChangeGradDate"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnChangeDate_Click()

    ' Change the specified graduation date in the MARINE table
    Application.SetOption "Confirm Action Queries", False
    DoCmd.OpenQuery "qryChangeGradDate"
    Application.SetOption "Confirm Action Queries", True

    ' Update the drop down list for MCRD Grad Date
    Me!cmbMCRDGradDate.RowSource = "SELECT DISTINCTROW
[qryMCTGradDate].[GradDate] FROM [qryMCTGradDate];"

    ' Force user to refresh data before performing another date change
    Me!cmbDays.Enabled = False
    Me!txtMCTGradDate.Enabled = False
    Me!btnEquals.Enabled = False

    ' Provide a message indicating the change was made
    MsgBox "Date change was successful!"

End Sub

Private Sub btnEquals_Click()

    Dim varDate As Variant
    Dim intDays As Integer

    intDays = Me.cmbDays
    varDate = Me.cmbMCRDGradDate

    If IsNull(varDate) = False And IsNull(intDays) = False Then

        ' Format this date for use in the DataAdd function
        varDate = Format(varDate, "####/##/##")

        ' Add 120 days to this date
        varDate = DataAdd("d", intDays, varDate)

        ' Change calendar to this date
        Me!ctlCalendar.Value = varDate

        ' Convert the new date back into the original format
        varDate = Format(varDate, "yyyymmdd")

        ' Place the new date in the "upper date bound" text box on the
calling form.
        Me.txtMCTGradDate = varDate

        ' Allows user to change the graduation date in the MARINE table
        Me!btnChangeDate.Enabled = True
    End If

End Sub

Private Sub cmbMCRDGradDate_AfterUpdate()

    Dim varDate As Variant

    varDate = Me.cmbMCRDGradDate
    Me!txtMCTGradDate.Value = " "

    If IsNull(varDate) = False Then

        ' Format this date for use in the DataAdd function
        varDate = Format(varDate, "####/##/##")

        ' Change calendar to this date
        Me!ctlCalendar.Value = varDate
        Me!ctlCalendar.Visible = True
        Me!cmbDays.Enabled = True
        Me!txtMCTGradDate.Enabled = True
        Me!btnEquals.Enabled = True

    End If

End Sub

Private Sub Form_Current()

    Me!ctlCalendar.Visible = False
    Me!cmbDays.Enabled = False
    Me!txtMCTGradDate.Enabled = False
    Me!btnEquals.Enabled = False
    Me!btnChangeDate.Enabled = False

End Sub
Private Sub Close_Click()
On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmImport/ExportSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub

Form: frmClass
    Code
Attribute VB_Name = "Form_frmClass"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnFindSchool_Click()
On Error GoTo Err_btnFindSchool_Click


    Screen.PreviousControl.SetFocus
    DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70

Exit_btnFindSchool_Click:
    Exit Sub

Err_btnFindSchool_Click:
    MsgBox Err.Description
    Resume Exit_btnFindSchool_Click

End Sub

Form: frmClassQuotaPenaltyAndFit
    Code
Attribute VB_Name = "Form_frmClassQuotaPenaltyAndFit"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnGo_Click()

    Dim varUDateBound
    Dim lngHour As Long, lngMin As Long, lngSec As Long
    Dim lngStart As Long, lngEnd As Long
    Dim strSQL As String

    ' Starts the timer for displaying the time taken for this subroutine
    lngStart = Timer


    ' Prevents the user from going back to change the MCTGradDate until a
fitness file is
    ' made, or the reset button is pushed.
    Me!MCTGradDate.Enabled = False
    varUDateBound = Me.UDateBound

    ' Generate the quota and penalty files
    Quota_Penalty (varUDateBound)

    ' Update the quota/penalty records on this form
    strSQL = "SELECT DISTINCTROW [CLASS].[SCourseNumber_FK], " & _
    "[CLASS].[AMOS_FK],[CLASS].[ClassNumber_PK], [CLASS].[ReportDate], "
    "[CLASS].[Quota], [CLASS].[ClassIndex], [PENALTY].[ClassIndex_FK], "
    "[PENALTY].[Penalty] FROM ([CLASS] INNER JOIN [PENALTY] ON " & _
    "[CLASS].[ClassIndex] =[PENALTY].[ClassIndex_FK]);"
    Me.RecordSource = strSQL

    ' update status of status text box
    Me!Status = "Working...."

    ' Call the Fitness procedure
    Fitness

    ' update status of status text box
    Me!Status = "Files generated!"
    DoCmd.Beep

    ' Records the stop time for running this subroutine
    lngEnd = Timer
    ' Determine total secs
    lngSec = lngEnd - lngStart
    ' Determine total hours
    lngHour = lngSec \ 3600
    ' Determine total mins
    lngMin = (lngSec - lngHour * 3600) \ 60
    ' Determine the secs remaining
    lngSec = (lngSec - (lngMin * 60) - (lngHour * 3600)) \ 1
    MsgBox "This operation took " & lngHour & " hour(s), " & lngMin & "
min(s), " & _
    "and " & lngSec & " secs.", vbOKOnly, "Recruit Distribution Model"

    Me.btnPrepareAndExecuteSolver.SetFocus


End Sub

Private Sub btnGo_MouseDown(Button As Integer, Shift As Integer, X As
Y As Single)

```

```
    Me!Status = "Working...."

End Sub

Private Sub btnPrepareAndExecuteSolver_Click()
On Error GoTo Err_btnPrepareAndExecuteSolver_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPrepareAndExecuteSolver"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnPrepareAndExecuteSolver_Click:
    Exit Sub

Err_btnPrepareAndExecuteSolver_Click:
    MsgBox Err.Description
    Resume Exit_btnPrepareAndExecuteSolver_Click

End Sub

Private Sub btnReturnPrevious_Click()
On Error GoTo Err_btnReturnPrevious_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPreprocessing&ExecutionSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnReturnPrevious_Click:
    Exit Sub

Err_btnReturnPrevious_Click:
    MsgBox Err.Description
    Resume Exit_btnReturnPrevious_Click

End Sub

Private Sub cmbCourseNumberFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SCourseNumber_FK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbCourseNumberFind] = Null
    Me.Penalty.SetFocus

End Sub

Private Sub Form_Current()

    ' Prevents the user from creating the AMPL files until a MCT
    ' graduation date is entered
    If IsNull(Me.MCTGradDate.Value) Then
        Me!btnGo.Enabled = False
        Me!Status.Enabled = False

    ' Allows the user to create the quota and penalty files
    Else
        Me!btnGo.Enabled = True

    End If
    ' update the status of the status text box
    Me!Status = " "

End Sub

Private Sub MCTGradDate_AfterUpdate()

    Dim varDate As Variant

    varDate = Me.MCTGradDate

    If IsNull(varDate) = False Then

        Me!btnGo.Enabled = True
        Me!Status.Enabled = True

        ' Format this date for use in the DateAdd function
        varDate = Format(varDate, "####/##/##")

        ' Add 120 days to this date
        varDate = DateAdd("d", 120, varDate)

        ' Convert the new date back into the original format
        varDate = Format(varDate, "yyyymmdd")

        ' Place the new date in the "upper date bound" text box on the
calling form.
        Me.UDateBound = varDate

    Else
        Me!btnGo.Enabled = False
        Me!Status.Enabled = False
    End If

End Sub


Form: frmFind
    Code
Attribute VB_Name = "Form_frmFind"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnFind_Click()
On Error GoTo Err_btnFind_Click

    'Screen.PreviousControl.SetFocus
    Forms!frmSchoolToPropertyMaint.SetFocus
    DoCmd.FindRecord Me.cmbCourseNumber, , True, , True, acAll
    'DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70

Exit_btnFind_Click:
    Exit Sub

Err_btnFind_Click:
    MsgBox Err.Description
    Resume Exit_btnFind_Click
End Sub



Form: frmFundamentalProperty
    Code
Attribute VB_Name = "Form_frmFundamentalProperty"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub Command15_Click()

    MsgBox "This is a button", vbInformation, "Testing"

End Sub

Private Sub btnClose_Click()
On Error GoTo Err_btnClose_Click


    DoCmd.Close
    DoCmd.OpenForm "frmMaintenanceSwitchboard"

Exit_btnClose_Click:
    Exit Sub

Err_btnClose_Click:
    MsgBox Err.Description
    Resume Exit_btnClose_Click

End Sub


Private Sub cmbFPropertyNameFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[FPropertyName_PK] = " & Chr(34) &
Me![cmbFPropertyNameFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbFPropertyNameFind] = Null
    Me.FPropertyName_PK.SetFocus

End Sub

Private Sub Command11_Click()
        ' Forces user to enter data for property name and description
    If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!MarineField.Enabled = False
        Me!Operator.Enabled = False
        Me!Label10.Visible = False
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = False
        Me!cmbPotentialValue.Visible = False
        MsgBox "You must provide a 'Property Name' and 'Description'
before building the fundamental equation."

    ' Allows the user to enter values for the fundamental equation
    Else
        Me!MarineField.Enabled = True
        Me!Operator.Enabled = True
        Me!Label10.Visible = True
        If Forms!frmFundamentalProperty.Operator.Value = "in" Then
            Me!subfrmFundamentalProperty.Visible = False
            Me!subfrmFundamentalPropertyList.Visible = True

        Else
            Me!cmbPotentialValue.Visible = True
            Me!subfrmFundementalProperty.Visible = True
            Me!subfrmFundamentalPropertyList.Visible = False
        End If
    End If

End Sub

Private Sub Description_AfterUpdate()

    Dim strSQL As String

        ' Forces user to enter data for property name and description
    If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!MarineField.Enabled = False
        Me!Operator.Enabled = False
        Me!Label10.Visible = False
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = False

    ' Allows the user to enter values for the fundamental equation
    Else
        Me!MarineField.Enabled = True
        Me!Operator.Enabled = True
        Me!Label10.Visible = True
        If Forms!frmFundamentalProperty.Operator.Value = "in" Then
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM

Forms!frmFundementalProperty!subfrmFundamentalPropertyList!Value_FK.RowSo
urce = strSQL
            Me!subfrmFundamentalProperty.Visible = False
            Me!subfrmFundamentalPropertyList.Visible = True

        Else
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM

Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSou
rce = strSQL
            Me!subfrmFundamentalProperty.Visible = True
            Me!subfrmFundamentalPropertyList.Visible = False
        End If
    End If

End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)

    [FundPropTimeStamp] = Now

End Sub

Private Sub Form_Current()

    Dim strSQL As String

    'strSQL = "SELECT MarineField, Operator FROM FUNDAMENTAL_PROPERTY
WHERE FPropertyName_PK = " & FundPropName


```

```
        ' Forces user to enter data for property name and description
    If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!MarineField.Enabled = False
        Me!Operator.Enabled = False
        Me!Label10.Visible = False
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = False

    ' Allows the user to enter values for the fundamental equation
    Else
        Me!MarineField.Enabled = True
        Me!Operator.Enabled = True
        Me!Label10.Visible = True
        If Forms!frmFundamentalProperty.Operator.Value = "In" Then
            ' Update the Potential Value list
            Me!subfrmFundamentalProperty.Visible = False
            Me!subfrmFundamentalPropertyList.Visible = True
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM

Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSource = strSQL
        Else
            ' Update the Potential Value list
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM

Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSource = strSQL
            Me!subfrmFundamentalProperty.Visible = True
            Me!subfrmFundamentalPropertyList.Visible = False
        End If

        ' Set focus to the property name. This prevents problem associated with entering the "Operator" object.
        FPropertyName_PK.SetFocus
    End If



End Sub

Private Sub FPropertyName_PK_AfterUpdate()

    Dim strSQL As String

        ' Forces user to enter data for property name and description
    If IsNull(Me.FPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!MarineField.Enabled = False
        Me!Operator.Enabled = False
        Me!Label10.Visible = False
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = False

    ' Allows the user to enter values for the fundamental equation
    Else
        Me!MarineField.Enabled = True
        Me!Operator.Enabled = True
        Me!Label10.Visible = True
        If Forms!frmFundamentalProperty.Operator.Value = "In" Then
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM

Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSource = strSQL
            Me!subfrmFundamentalProperty.Visible = False
            Me!subfrmFundamentalPropertyList.Visible = True

        Else
            strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM

Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSource = strSQL
            Me!subfrmFundamentalProperty.Visible = True
            Me!subfrmFundamentalPropertyList.Visible = False
        End If
    End If

End Sub

Private Sub MarineField_AfterUpdate()

    Dim strSQL As String

    ' Update the Potential Value list
    strSQL = "SELECT DISTINCT " & Me!MarineField & " FROM MARINE;"

Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK.RowSource = strSQL

Forms!frmFundamentalProperty!subfrmFundamentalProperty!cmbValue_FK.RowSource = strSQL

End Sub

Private Sub Operator_AfterUpdate()

    If Forms!frmFundamentalProperty.Operator.Value = "In" Then
        Me!subfrmFundamentalProperty.Visible = False
        Me!subfrmFundamentalPropertyList.Visible = True
    Else
        Me!subfrmFundamentalProperty.Visible = True
        Me!subfrmFundamentalPropertyList.Visible = False
    End If

End Sub
Private Sub btnDelete_Click()
On Error GoTo Err_btnDelete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_btnDelete_Click:
    Exit Sub

Err_btnDelete_Click:
    MsgBox Err.Description
    Resume Exit_btnDelete_Click

End Sub
Private Sub Command14_Click()
On Error GoTo Err_Command14_Click

    Dim stDialStr As String
    Dim PrevCtl As Control
    Const ERR_OBJNOTEXIST = 2467
    Const ERR_OBJNOTSET = 91

    Set PrevCtl = Screen.PreviousControl

    If TypeOf PrevCtl Is TextBox Then
      stDialStr = IIf(VarType(PrevCtl) > V_NULL, PrevCtl, "")
    ElseIf TypeOf PrevCtl Is ListBox Then
      stDialStr = IIf(VarType(PrevCtl) > V_NULL, PrevCtl, "")
    ElseIf TypeOf PrevCtl Is ComboBox Then
      stDialStr = IIf(VarType(PrevCtl) > V_NULL, PrevCtl, "")
    Else
      stDialStr = ""
    End If

    Application.Run "utility.wlib_AutoDial", stDialStr

Exit_Command14_Click:
    Exit Sub

Err_Command14_Click:
    If (Err = ERR_OBJNOTEXIST) Or (Err = ERR_OBJNOTSET) Then
      Resume Next
    End If
    MsgBox Err.Description
    Resume Exit_Command14_Click

End Sub

Private Sub Operator_Enter()

    If Me.Operator.Value = "In" Then
        MsgBox "The 'In' operator is not changeable. It is removed by deleting the property.", vbInformation, "Unauthorized Action"
        subfrmFundamentalPropertyList.SetFocus
    End If

End Sub

Form: frmGenerateReport
    Code
    Attribute VB_Name = "Form_frmGenerateReport"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnAssignmentReport_Click()

        DoCmd.OpenReport "rptassignedMarines", acViewPreview

    End Sub

    Private Sub btnEntryLevelSchool_Click()

        DoCmd.OpenReport "rptEntryLevelSchool", acViewPreview


    End Sub

    Private Sub btnExit_Click()
    On Error GoTo Err_btnExit_Click


        DoCmd.Close

    Exit_btnExit_Click:
        Exit Sub

    Err_btnExit_Click:
        MsgBox Err.Description
        Resume Exit_btnExit_Click


    End Sub

    Private Sub btnUnassignedMarineReport_Click()

        DoCmd.OpenReport "rptUnassignedMarines", acViewPreview

    End Sub

    Private Sub Command8_Click()
    On Error GoTo Err_Command8_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmRDM_Main_Switchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command8_Click:
        Exit Sub

    Err_Command8_Click:
        MsgBox Err.Description
        Resume Exit_Command8_Click

    End Sub

Form: frmImport/ExportSwitchboard
    Code
    Attribute VB_Name = "Form_frmImport/ExportSwitchboard"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnExit_Click()
    On Error GoTo Err_btnExit_Click


        DoCmd.Close

    Exit_btnExit_Click:
        Exit Sub

    Err_btnExit_Click:
        MsgBox Err.Description
        Resume Exit_btnExit_Click


    End Sub

    Private Sub btnExportRD3_Click()

        Application.SetOption "Confirm Action Queries", False
        DoCmd.SetWarnings (False)
        ' Exports assignments to RD3.txt file
        ExportRD3
        ' Archives data from the Marine and assignment tables, having report dates less than the current date
    ArchiveData
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

End Sub


Private Sub Command2_Click()

    Dim stDocName As String
    Dim stLinkCriteria As String

    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)
    ' Imports the RD1 file
    ImportRD1
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmChangeGradDate"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

End Sub

Private Sub Command4_Click()

    ' Imports BNA.txt file
    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)
    DoCmd.RunSQL "DELETE * FROM BNA_EXTRACT;"
    ImportBNA
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

End Sub

Private Sub Command8_Click()

On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmRDM_Main_Switchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

Err_Command8_Click:
    MsgBox Err.Description
    Resume Exit_Command8_Click

End Sub

Form: frmLogicalProperty
  Code
Attribute VB_Name = "Form_frmLogicalProperty"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnClose_Click()
On Error GoTo Err_btnClose_Click


    DoCmd.Close
    DoCmd.OpenForm "frmMaintenanceSwitchboard"

Exit_btnClose_Click:
    Exit Sub

Err_btnClose_Click:
    MsgBox Err.Description
    Resume Exit_btnClose_Click

End Sub

Private Sub cmbFundProp_Click()

    ' Adds the selected fundamental property to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value &
Me.cmbFundProp.Value & " "

End Sub
Private Sub cmbLogOper_Click()

    ' Adds the selected logical operator to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value &
Me.cmbLogOper.Value & " "

End Sub

Private Sub cmbLPropertyNameFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[LPropertyName_PK] = " & Chr(34) &
Me![cmbLPropertyNameFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbLPropertyNameFind] = Null
    Me.LPropertyName_PK.SetFocus

End Sub

Private Sub Command10_Click()

    ' Removes the last entry in the logical equation

    Dim strInName, strUndo  As String        ' Input name
    Dim intSpacePos         As Integer       ' position of the space
    Dim I                   As Integer       ' loop index
    Dim booResult           As Boolean
    Dim lngConversion       As Long
    Dim varConv             As Variant


    ' get the logical expression
    strInName = Me.LogicalEquation.Value

    'Check for "Null" value
    If strInName <> "Null" Then

        ' declare the temporary string as an array
        ReDim strTemp(0) As String

        ' Find the position of the first space
        intSpacePos = InStr(strInName, " ")

        ' Loop until no more spaces are found. InStr returns 0 if the
space is not found.
        Do Until intSpacePos = 0
            ' redimension the array, adding another element
            ' UBound tells us how many elements there already are in the
            ReDim Preserve strTemp(UBound(strTemp) + 1)

            ' set the new element to the characters from the beginning of
            ' string up to the first space.  Left$ returns a number of
characters
            ' from the beginning of a string
            strTemp(UBound(strTemp)) = Trim$(Left$(strInName, intSpacePos

            ' now we've copied these characters into the name, they can
            ' removed from the original string.  Right$ returns a number
            ' characters from the end of the string.  It is the rightmost
            ' characters, therefore we subtract the position of the space
            ' the length of the string.  This gives us the number of
characters remaining.
            strInName = LTrim$(Right$(strInName, Len(strInName) -

            ' find the next space
            intSpacePos = InStr(strInName, " ") 'Or intSpacePos =
InStr(strInName, "(") 'Or intSpacePos = InStr(strInName, ")")

        Loop

        ' Remove the last element of the array, unless there is only one
element left
        If UBound(strTemp) > 1 Then
            ReDim Preserve strTemp(UBound(strTemp) - 1)

            ' Loop through the elements of the array and place them back
into the logical equation expression
            For I = 1 To (UBound(strTemp))
              strUndo = strUndo & strTemp(I) & " "
              Me.LogicalEquation.Value = strUndo
            Next I
        Else
            ' Set the logical equation equal to "Null"
            Me.LogicalEquation.Value = "( "
        End If

        ' Set logical equation expression equal to new value
    End If

End Sub

Private Sub Command11_Click()

    ' Forces user to enter data for property name and description
    If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!Command8.Enabled = False
        Me!Command9.Enabled = False
        Me!Command10.Enabled = False
        Me!cmbFundProp.Enabled = False
        Me!cmbLogOper.Enabled = False
        Me!LogicalEquation.Enabled = False
        MsgBox "You must provide a 'Property Name' and 'Description'
before building the logical equation."

    ' Enters a "( " be the first entry in the logical equation if it is
    ElseIf IsNull(Me!LogicalEquation.Value) Then
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
        Me!LogicalEquation.Value = "( "

    ' Allows the user to enter values into the logical equation
    Else
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
    End If

End Sub

Private Sub Command8_Click()

    ' Adds the left bracket to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value & "( "

End Sub
Private Sub Command9_Click()

    ' Adds the right bracket to the logical equation
    Me.LogicalEquation.Value = Me.LogicalEquation.Value & ") "

End Sub
Private Sub btnDelete_Click()
On Error GoTo Err_btnDelete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_btnDelete_Click:
    Exit Sub

Err_btnDelete_Click:
    MsgBox Err.Description
    Resume Exit_btnDelete_Click

End Sub
Private Sub Description_AfterUpdate()

    ' Forces user to enter data for property name and description
    If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!Command8.Enabled = False
        Me!Command9.Enabled = False
        Me!Command10.Enabled = False
        Me!cmbFundProp.Enabled = False
        Me!cmbLogOper.Enabled = False
        Me!LogicalEquation.Enabled = False

    ' Enters a "( " be the first entry in the logical equation if it is
    ElseIf IsNull(Me!LogicalEquation.Value) Then
```

```
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
        Me!LogicalEquation.Value = "( "

    ' Allows the user to enter values into the logical equation
    Else
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
    End If

End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)

    [LogPropTimeStamp] = Now

End Sub

Private Sub Form_Current()

    ' Forces user to enter data for property name and description
    If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!Command8.Enabled = False
        Me!Command9.Enabled = False
        Me!Command10.Enabled = False
        Me!cmbFundProp.Enabled = False
        Me!cmbLogOper.Enabled = False
        Me!LogicalEquation.Enabled = False

    ' Enters a "( " be the first entry in the logical equation if it is
    ElseIf IsNull(Me!LogicalEquation.Value) Then
        Me!LogicalEquation.Value = "( "

    ' Allows the user to enter values into the logical equation
    Else
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
    End If

End Sub
Private Sub LogicalEquation_Enter()

    ' Prevents the user from modifing the logical equation directly
    MsgBox "The logical equation is only modifiable by the above logical
equation builder"
    DoCmd.GoToControl "LPropertyName_PK"

End Sub

Private Sub LPropertyName_PK_AfterUpdate()

    ' Forces user to enter data for property name and description
    If IsNull(Me.LPropertyName_PK.Value) Or IsNull(Me.Description.Value)
        Me!Command8.Enabled = False
        Me!Command9.Enabled = False
        Me!Command10.Enabled = False
        Me!cmbFundProp.Enabled = False
        Me!cmbLogOper.Enabled = False
        Me!LogicalEquation.Enabled = False

    ' Enters a "( " be the first entry in the logical equation if it is
    ElseIf IsNull(Me!LogicalEquation.Value) Then
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
        Me!LogicalEquation.Value = "( "

    ' Allows the user to enter values into the logical equation
    Else
        Me!Command8.Enabled = True
        Me!Command9.Enabled = True
        Me!Command10.Enabled = True
        Me!cmbFundProp.Enabled = True
        Me!cmbLogOper.Enabled = True
        Me!LogicalEquation.Enabled = True
    End If

End Sub

Form: frmMaintenanceSwitchboard
    Code
Attribute VB_Name = "Form_frmMaintenanceSwitchboard"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnExit_Click()
On Error GoTo Err_btnExit_Click


    DoCmd.Close

Exit_btnExit_Click:
    Exit Sub

Err_btnExit_Click:
    MsgBox Err.Description
    Resume Exit_btnExit_Click


End Sub

Private Sub Command0_Click()
On Error GoTo Err_Command0_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmSchoolToPEF_Maint"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command0_Click:
    Exit Sub

Err_Command0_Click:
    MsgBox Err.Description
    Resume Exit_Command0_Click

End Sub

Private Sub Command12_Click()
On Error GoTo Err_Command12_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmSchools"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command12_Click:
    Exit Sub

Err_Command12_Click:
    MsgBox Err.Description
    Resume Exit_Command12_Click

End Sub

Private Sub Command14_Click()
On Error GoTo Err_Command14_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPEF"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command14_Click:
    Exit Sub

Err_Command14_Click:
    MsgBox Err.Description
    Resume Exit_Command14_Click

End Sub

Private Sub Command2_Click()
On Error GoTo Err_Command2_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmFundamentalProperty"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command2_Click:
    Exit Sub

Err_Command2_Click:
    MsgBox Err.Description
    Resume Exit_Command2_Click

End Sub

Private Sub Command4_Click()
On Error GoTo Err_Command4_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmLogicalProperty"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command4_Click:
    Exit Sub

Err_Command4_Click:
    MsgBox Err.Description
    Resume Exit_Command4_Click

End Sub

Private Sub Command6_Click()
On Error GoTo Err_Command6_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmSchoolToPropertyMaint"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command6_Click:
    Exit Sub

Err_Command6_Click:
    MsgBox Err.Description
    Resume Exit_Command6_Click

End Sub

Private Sub Command8_Click()
On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmRDM_Main_Switchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

    Err_Command8_Click:
        MsgBox Err.Description
        Resume Exit_Command8_Click

    End Sub

Form: frmPEF
  Code
    Attribute VB_Name = "Form_frmPEF"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub cmbPefFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[PEF_PK] = " & Chr(34) & Me![cmbPefFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbPefFind] = Null
        Me.PEF_PK.SetFocus

    End Sub

    Private Sub Form_BeforeUpdate(Cancel As Integer)

        [PEF_TimeStamp] = Now

    End Sub

    Private Sub Form_Current()

        Dim strSQL As String

        ' Update the PEF List
        strSQL = "SELECT DISTINCTROW [MARINE].[PEF]FROM MARINE LEFT JOIN PEF
    ON [MARINE].[PEF] = [PEF].[PEF_PK]WHERE ([PEF].[PEF_PK] Is Null)ORDER BY
        Me.PEF_PK.RowSource = strSQL

    End Sub

    Private Sub PEF_PK_AfterUpdate()

        Dim strSQL As String

        ' Update the PEF List
        strSQL = "SELECT DISTINCTROW [MARINE].[PEF]FROM MARINE LEFT JOIN PEF
    ON [MARINE].[PEF] = [PEF].[PEF_PK]WHERE ([PEF].[PEF_PK] Is Null)ORDER BY
        Me.PEF_PK.RowSource = strSQL

    End Sub
    Private Sub btnDelete_Click()
    On Error GoTo Err_btnDelete_Click


        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_btnDelete_Click:
        Exit Sub

    Err_btnDelete_Click:
        MsgBox Err.Description
        Resume Exit_btnDelete_Click

    End Sub
    Private Sub btnClose_Click()
    On Error GoTo Err_btnClose_Click


        DoCmd.Close
        DoCmd.OpenForm "frmMaintenanceSwitchboard"

    Exit_btnClose_Click:
        Exit Sub

    Err_btnClose_Click:
        MsgBox Err.Description
        Resume Exit_btnClose_Click

    End Sub

Form: frmPrepareAndExecuteSolver
  Code
    Attribute VB_Name = "Form_frmPrepareAndExecuteSolver"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnAMPL_Click()
    On Error GoTo Err_btnAMPL_Click

        AmplData

        Dim stAppName As String

        'stAppName = "C:\ToBeRdm\Amplwin\amplwin.exe"
        stAppName = "C:\ToBeRdm\Amplwin\amp.bat"
        Call Shell(stAppName, 0)

        'MsgBox "Enter into the command line: 'include rdm.run'"

    Exit_btnAMPL_Click:
        Exit Sub

    Err_btnAMPL_Click:
        MsgBox Err.Description
        Resume Exit_btnAMPL_Click

    End Sub

    Private Sub btnAmplPlus_Click()
    On Error GoTo Err_btnAmplPlus_Click

        AmplData

        Dim stAppName As String

        stAppName = "C:\AMPLPLUS\AMPLPLUS.EXE rdm.amp"
        Call Shell(stAppName, 1)

    Exit_btnAmplPlus_Click:
        Exit Sub

    Err_btnAmplPlus_Click:
        MsgBox Err.Description
        Resume Exit_btnAmplPlus_Click

    End Sub


    Private Sub btnReturnPrevious_Click()
    On Error GoTo Err_btnReturnPrevious_Click


        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmPreprocessing&ExecutionSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_btnReturnPrevious_Click:
        Exit Sub

    Err_btnReturnPrevious_Click:
        MsgBox Err.Description
        Resume Exit_btnReturnPrevious_Click


    End Sub

    Private Sub Command4_Click()
        Application.SetOption "Confirm Action Queries", False
        DoCmd.SetWarnings (False)
        DoCmd.RunSQL "DELETE * FROM AMPL_RESULT;"
        Ampl_Result
        Application.SetOption "Confirm Action Queries", True
        DoCmd.SetWarnings (True)

    End Sub

    Private Sub Command7_Click()
    On Error GoTo Err_Command7_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmAnalyzeResult"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Command7_Click:
        Exit Sub

    Err_Command7_Click:
        MsgBox Err.Description
        Resume Exit_Command7_Click

    End Sub

Form: frmPreprocessing&ExecutionSwitchboard
  Code
    Attribute VB_Name = "Form_frmPreprocessing&ExecutionSwitchboard"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnExit_Click()
    On Error GoTo Err_btnExit_Click


        DoCmd.Close

    Exit_btnExit_Click:
        Exit Sub

    Err_btnExit_Click:
        MsgBox Err.Description
        Resume Exit_btnExit_Click


    End Sub

    Private Sub btnReclassification_Click()
    On Error GoTo Err_btnReclassification_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmReclassification"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_btnReclassification_Click:
        Exit Sub

    Err_btnReclassification_Click:
        MsgBox Err.Description
        Resume Exit_btnReclassification_Click


    End Sub

    Private Sub btnScrubMarineData_Click()
    On Error GoTo Err_btnScrubMarineData_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close the current form
        DoCmd.Close
        ' Open specified form
        stDocName = "frmScrubMarine"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_btnScrubMarineData_Click:
        Exit Sub

    Err_btnScrubMarineData_Click:
        MsgBox Err.Description
        Resume Exit_btnScrubMarineData_Click

    End Sub

    Private Sub btnSpecialAssignment_Click()
    On Error GoTo Err_btnSpecialAssignment_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmSpecialAssignment"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_btnSpecialAssignment_Click:
    Exit Sub

Err_btnSpecialAssignment_Click:
    MsgBox Err.Description
    Resume Exit_btnSpecialAssignment_Click



End Sub

Private Sub Command2_Click()
On Error GoTo Err_Command2_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmclassQuotaPenaltyAndFit"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command2_Click:
    Exit Sub

Err_Command2_Click:
    MsgBox Err.Description
    Resume Exit_Command2_Click

End Sub
Private Sub Command4_Click()

    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)
    DoCmd.RunSQL "DELETE * FROM AMPL_RESULT;"
    Ampl_Result
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)


End Sub

Private Sub Command6_Click()
On Error GoTo Err_Command6_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPrepareAndExecuteSolver"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command6_Click:
    Exit Sub

Err_Command6_Click:
    MsgBox Err.Description
    Resume Exit_Command6_Click

End Sub

Private Sub Command8_Click()
On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmRDM_Main_Switchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

Err_Command8_Click:
    MsgBox Err.Description
    Resume Exit_Command8_Click

End Sub

Form: frmRDM_Main_Switchboard
    Code
Attribute VB_Name = "Form_frmRDM_Main_Switchboard"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit
Private Sub Command0_Click()

On Error GoTo Err_Command0_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmImport/ExportSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command0_Click:
    Exit Sub

Err_Command0_Click:
    MsgBox Err.Description
    Resume Exit_Command0_Click

End Sub
Private Sub Command2_Click()
On Error GoTo Err_Command2_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmMaintenanceSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command2_Click:
    Exit Sub

Err_Command2_Click:
    MsgBox Err.Description
    Resume Exit_Command2_Click

End Sub
Private Sub Command4_Click()
On Error GoTo Err_Command4_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmPreprocessing&ExecutionSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command4_Click:
    Exit Sub

Err_Command4_Click:
    MsgBox Err.Description
    Resume Exit_Command4_Click

End Sub
Private Sub Command6_Click()
On Error GoTo Err_Command6_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmAnalyzeResult"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command6_Click:
    Exit Sub

Err_Command6_Click:
    MsgBox Err.Description
    Resume Exit_Command6_Click

End Sub
Private Sub Command8_Click()
On Error GoTo Err_Command8_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmGenerateReport"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Command8_Click:
    Exit Sub

Err_Command8_Click:
    MsgBox Err.Description
    Resume Exit_Command8_Click

End Sub
Private Sub btnExit_Click()
On Error GoTo Err_btnExit_Click


    DoCmd.Close

Exit_btnExit_Click:
    Exit Sub

Err_btnExit_Click:
    MsgBox Err.Description
    Resume Exit_btnExit_Click

End Sub

Form: frmReclassification
    Code
Attribute VB_Name = "Form_frmReclassification"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub cmbClassNumber_AfterUpdate()

    Dim strConvert As String

    Me.txtReportDate = Me.cmbClassNumber.Column(2)
    Me.txtMCC = Me.cmbClassNumber.Column(4)

    ' Convert the fiscal year to a two digit number for RD3 File
    strConvert = CStr(Me.cmbClassNumber.Column(3))
    strConvert = Right(strConvert, 2)
    Me.txtFY = CInt(strConvert)

    Me.txtAssignmentType = "R"

End Sub

Private Sub cmbClassNumber_Enter()

    Dim strSQL As String

    ' This query finds the class numbers, report dates and class
convening dates associated with the chosen school
    strSQL = "SELECT BNA_EXTRACT.ClassNumber_PK,
BNA_EXTRACT.ClassConvDate, BNA_EXTRACT.ReportDate,
BNA_EXTRACT.FiscalYear_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT INNER JOIN
SCH_TGT_MOS ON (BNA_EXTRACT.CourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK) AND (BNA_EXTRACT.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK) WHERE BNA_EXTRACT.CourseNumber_PK =
Forms!frmReclassification!cmbCourseNumber AND SCH_TGT_MOS.AMOS_FK =
Forms!frmReclassification!txtAMOS;"
    Me.cmbClassNumber.RowSource = strSQL

End Sub
```

```

Private Sub cmbCourseNumber_AfterUpdate()

    Me.txtAMOS = Me!cmbCourseNumber.Column(1)

End Sub

Private Sub cmbSSN_AfterUpdate()

    Me.txtSOI = Me!cmbSSN.Column(1)
    Me.txtGradDate = Me!cmbSSN.Column(2)

End Sub

Private Sub cmbSSN_Enter()

    Dim strSQL As String

    strSQL = "SELECT DISTINCTROW MARINE.SSN_PK, MARINE.SOI,
MARINE.GradDate FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK =
ASSIGNMENT.SSN_FK WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
    Me.cmbSSN.RowSource = strSQL

End Sub

Private Sub cmbSSNFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbSSNFind] = Null
    Me.cmbSSN.SetFocus

End Sub

Private Sub Delete_Click()
On Error GoTo Err_Delete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_Delete_Click:
    Exit Sub

Err_Delete_Click:
    MsgBox Err.Description
    Resume Exit_Delete_Click

End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close current form
    DoCmd.Close

    ' Open specified form
    stDocName = "frmPreprocessing&ExecutionSwitchboard"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub


Form: frmSchoolAssignments
    Code
Attribute VB_Name = "Form_frmSchoolAssignments"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit
Private Sub Close_Click()
On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String
    Dim db1 As Database

    Set db1 = CurrentDb()

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmAnalyzeResult"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
    Forms!frmAnalyzeResult.SetFocus


Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub

Private Sub Form_Current()

    Dim strSQL As String
    Dim rec As Recordset
    Dim db1 As Database

    Set db1 = CurrentDb()

    ' Calculates the total quota, and number of schools for the run.
    strSQL = "SELECT Sum(SumOfQuota) AS TotalQuota, Count(SumOfQuota) AS
NumberOfSchools FROM qryTotalQuotaForRun;"
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtTotalQuota = rec!TotalQuota
    Me.txtNumberOfSchools = rec!NumberOfSchools

    ' Calculates the total number of assignements for the run.
    strSQL = "SELECT Sum(CountOfFiscalYear_PK) AS TotalAssigned FROM
qryTotalQuotaFilledForRun;"
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtTotalAssigned = rec!TotalAssigned

    Me.txtTotalPercentFill = Int((Me.txtTotalAssigned / Me.txtTotalQuota)


End Sub

Form: frmSchools
    Code
Attribute VB_Name = "Form_frmSchools"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub AMOS_PK_AfterUpdate()
    Dim rec As Recordset
    Dim db1 As Database
    Dim strSQL As String

    ' Update the drop down list for AMOS
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    'Me.AMOS_PK.RowSource = strSQL
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the AMOS_FK in the child relation SCH_TGT_MOS
    'Forms!frmSchools!subfrmSchools!AMOS_FK.Value = Me.AMOS_PK.Value

    ' Update the TargetMOS_FK in the child relation SCH_TGT_MOS
    'Set db1 = CurrentDb()
    'Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    'Forms!frmSchools!subfrmSchools!TargetMOS_FK.Value = rec!TargetMOS_PK

    ' Input the correct MCC value for the selected course
    strSQL = "SELECT MCC FROM TARGET_MOS WHERE TCourseNumber_PK = '" &
Me.SCourseNumber_PK.Value & "'"
    Set db1 = CurrentDb()
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.MCC.Value = rec!MCC

    ' Update the TCourseNumber_FK in the child relation SCH_TGT_MOS
    'Forms!frmSchools!subfrmSchools!TCourseNumber_FK.Value =
Me.SCourseNumber_PK.Value

End Sub


Private Sub btnClear_Click()

    Dim db1 As Database
    Dim strSQL As String
    Dim i As Integer
    Dim rec As Recordset

    Set db1 = CurrentDb()

    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null)AND((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
TCourseNumber_PK;"
    Set rec = db1.OpenRecordset(strSQL, dbOpenDynaset)

    If rec.EOF = False Then
        rec.MoveLast
        rec.MoveFirst
    End If

    ' Clear unused course numbers from the target_mos table

    If rec.RecordCount > 0 Then
        For i = 1 To rec.RecordCount
            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False
            DoCmd.SetWarnings (False)

            strSQL = "DELETE * FROM TARGET_MOS WHERE
TARGET_MOS.TCourseNumber_PK = '" & rec!TCourseNumber_PK & "'"
            db1.Execute strSQL

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True
            DoCmd.SetWarnings (True)
            rec.MoveNext
        Next i
    End If

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null)AND((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL

End Sub

Private Sub btnResetAllPenalties_Click()


    ' Disables the action query confirmation message
    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)

    DoCmd.RunSQL "UPDATE SCHOOL SET PenaltyFactor = 24;"

    ' Enables the action query confirmation message
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

    cmbPenaltyView = "1x"

End Sub

Private Sub cmbCourseNumberFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbCourseNumberFind] = Null
    Me.SCourseNumber_PK.SetFocus

End Sub

Private Sub cmbPenaltyView_AfterUpdate()

```

```
    Select Case cmbPenaltyView
        Case "4x"
            bdPenaltyFactor = 4 * 24
        Case "3x"
            bdPenaltyFactor = 3 * 24
        Case "2x"
            bdPenaltyFactor = 2 * 24
        Case "1x"
            bdPenaltyFactor = 1 * 24
        Case "1/2x"
            bdPenaltyFactor = 24 / 2
        Case "1/3x"
            bdPenaltyFactor = 24 / 3
        Case "1/4x"
            bdPenaltyFactor = 24 / 4
    End Select

    ' Save the update
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, ,

End Sub

Private Sub Command14_Click()

    Dim strSQL As String

    ' Disables the action query confirmation message
    Application.SetOption "Confirm Action Queries", False
    DoCmd.SetWarnings (False)

    DoCmd.RunSQL "INSERT INTO TARGET_MOS ( TCourseNumber_PK,
TargetMOS_PK, MCC ) SELECT DISTINCT BNA_EXTRACT.CourseNumber_PK,
BNA_EXTRACT.TargetMOS_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT;"

    ' Enables the action query confirmation message
    Application.SetOption "Confirm Action Queries", True
    DoCmd.SetWarnings (True)

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null)AND((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL

End Sub


Private Sub Command39_Click()
    Dim frm As Form, ctl As Control
    Dim varItm As Variant, intI As Integer
    Dim strSQL As String

    ' Ensures the current record is save to the SCHOOL table
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, ,

    ' This code enters the selected Target MOS's into the SCH_TGT_MOS
    Set frm = Forms!frmSchools
    Set ctl = frm!lstTargetMOS_FK
    For Each varItm In ctl.ItemsSelected
        For intI = 0 To ctl.ColumnCount - 1

            ' This puts the value found in the list box into a text box,
making it readable by the query
            Forms!frmSchools!TargetValue.Value = ctl.Column(intI, varItm)

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False

            ' Enters the value into the SCH_TGT_MOS table
            DoCmd.OpenQuery "qryUpdateSCH_TGT_MOS"

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True

        Next intI
    Next varItm

    ' Update the Target MOS list
    strSQL = "SELECT [TargetMOS_FK] FROM [SCH_TGT_MOS] WHERE
[SCourseNumber_FK] = '" & Forms!frmSchools!SCourseNumber_PK.Value & "'" &
" AND AMOS_FK = '" & Forms!frmSchools!AMOS_PK.Value & "'"
    Forms!frmSchools!ListTargetMOS_FK_History.RowSource = strSQL

    ' Update the Potential Target MOS list
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
    Me.SCourseNumber_PK.RowSource = strSQL

    ' Update time stamp
    [Sch_TimeStamp] = Now


End Sub

Private Sub Command40_Click()
    Dim frm As Form, ctl As Control
    Dim varItm As Variant, intI As Integer
    Dim strSQL As String

    ' This code Delets the selected Target MOS's and its associated
entries from the SCH_TGT_MOS table
    Set frm = Forms!frmSchools
    Set ctl = frm!ListTargetMOS_FK_History
    For Each varItm In ctl.ItemsSelected
        For intI = 0 To ctl.ColumnCount - 1

            ' This puts the value found in the list box into a text box,
making it readable by the query
            Forms!frmSchools!TargetValue.Value = ctl.Column(intI, varItm)

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False

            ' Deletes the record associated with the specified value from
the SCH_TGT_MOS table
            DoCmd.OpenQuery "qryDeleteSCH_TGT_MOS"

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True

        Next intI
    Next varItm

    ' Update the Target MOS list
    strSQL = "SELECT [TargetMOS_FK] FROM [SCH_TGT_MOS] WHERE
[SCourseNumber_FK] = '" & Forms!frmSchools!SCourseNumber_PK.Value & "'" &
" AND AMOS_FK = '" & Forms!frmSchools!AMOS_PK.Value & "'"
    Forms!frmSchools!ListTargetMOS_FK_History.RowSource = strSQL

    ' Update the Potential Target MOS list
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
    Me.SCourseNumber_PK.RowSource = strSQL

    ' Update time stamp
    [Sch_TimeStamp] = Now

End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)

    [Sch_TimeStamp] = Now

End Sub

Private Sub Form_Current()
    Dim strSQL As String

    ' Update the Potential Target MOS list
    strSQL = "SELECT TargetMOS_PK FROM
qryListUnusedTargetMOSFromTARGET_MOS WHERE TCourseNumber_PK = '" &
    Me.lstTargetMOS_FK.RowSource = strSQL

    ' Update the Target MOS List
    strSQL = "SELECT [TargetMOS_FK] FROM [SCH_TGT_MOS] WHERE
[SCourseNumber_FK] = '" & Me.SCourseNumber_PK.Value & "'" & " AND AMOS_FK
= '" & Me.AMOS_PK.Value & "'"
    Me.ListTargetMOS_FK_History.RowSource = strSQL

    ' Update the Course Number List
    strSQL = "SELECT DISTINCTROW TARGET_MOS.TCourseNumber_PK FROM
TARGET_MOS LEFT JOIN SCH_TGT_MOS ON TARGET_MOS.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK AND TARGET_MOS.TCourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK WHERE (((SCH_TGT_MOS.TargetMOS_FK) Is
Null)AND((SCH_TGT_MOS.TCourseNumber_FK) Is Null))ORDER BY
TCourseNumber_PK;"
    Me.SCourseNumber_PK.RowSource = strSQL

    ' Forces user to enter data for course number, before the Assigned
    If IsNull(Me.SCourseNumber_PK.Value) Then
        Me!AMOS_PK.Enabled = False

    ' Allows the user to enter an assigned MOS
    Else
        Me!AMOS_PK.Enabled = True
    End If

    ' Prevents the user from accidentally changing a course number
    If IsNull(SCourseNumber_PK) Then
        Me!SCourseNumber_PK.Locked = False
    Else
        Me!SCourseNumber_PK.Locked = True
    End If

        ' Prevents the user from accidentally changing an AMOS
    If IsNull(AMOS_PK) Then
        Me!AMOS_PK.Locked = False
    Else
        Me!AMOS_PK.Locked = True
    End If

    ' Update the penalty factor view
    Select Case bdPenaltyFactor
        Case 96
            cmbPenaltyView = "4x"
        Case 72
            cmbPenaltyView = "3x"
        Case 48
            cmbPenaltyView = "2x"
        Case 24
            cmbPenaltyView = "1x"
        Case 12
            cmbPenaltyView = "1/2x"
        Case 8
            cmbPenaltyView = "1/3x"
        Case 6
            cmbPenaltyView = "1/4x"
    End Select




End Sub

Private Sub SCourseNumber_PK_AfterUpdate()

    Dim rec As Recordset
    Dim db1 As Database
    Dim strSQL As String

    ' Update the drop down list for AMOS
    strSQL = "SELECT TargetMOS_PK FROM TARGET_MOS WHERE TCourseNumber_PK
= '" & Me.SCourseNumber_PK.Value & "'"
    Me.AMOS_PK.RowSource = strSQL
    'Me.lstTargetMOS_FK.RowSource = strSQL

    ' Allows the user to enter an assigned MOS
    Me!AMOS_PK.Enabled = True

End Sub
Private Sub btnDelete_Click()
On Error GoTo Err_btnDelete_Click

    ' Disables the action query confirmation message
    Application.SetOption "Confirm Action Queries", False

    ' Deletes the record associated with the specified value from the
SCH_TGT_MOS table
    DoCmd.OpenQuery "qryDeleteSCH_TGT_MOSEntry"

    ' Deletes the record associated with the specified value from the
SCHOOL table
    'DoCmd.OpenQuery "qryDeleteSCHOOLEntry"

    ' Enables the action query confirmation message
    Application.SetOption "Confirm Action Queries", True

    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_btnDelete_Click:
    Exit Sub

Err_btnDelete_Click:
    MsgBox Err.Description
    Resume Exit_btnDelete_Click

End Sub
Private Sub btnClose_Click()
On Error GoTo Err_btnClose_Click


    DoCmd.Close
    DoCmd.OpenForm "frmMaintenanceSwitchboard"

Exit_btnClose_Click:
    Exit Sub

Err_btnClose_Click:
    MsgBox Err.Description
    Resume Exit_btnClose_Click

End Sub

Form: frmSchoolToPEF_Maint
    Code
    Attribute VB_Name = "Form_frmSchoolToPEF_Maint"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnClose_Click()
    On Error GoTo Err_btnClose_Click


        DoCmd.Close
        DoCmd.OpenForm "frmMaintenanceSwitchboard"

    Exit_btnClose_Click:
        Exit Sub

    Err_btnClose_Click:
        MsgBox Err.Description
        Resume Exit_btnClose_Click


    End Sub

    Private Sub btnDelete_Click()
    On Error GoTo Err_btnDelete_Click


        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_btnDelete_Click:
        Exit Sub

    Err_btnDelete_Click:
        MsgBox Err.Description
        Resume Exit_btnDelete_Click

    End Sub

    Private Sub cmbCourseNumberFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbCourseNumberFind] = Null
        Me.SCourseNumber_PK.SetFocus

    End Sub

    Private Sub Form_Current()
        Dim strSQL As String

        ' Update the Potential PEF list
        strSQL = "SELECT DISTINCTROW [PEF].[PEF_PK] FROM PEF LEFT JOIN
qrySch_Pef ON [PEF].[PEF_PK] = [qrySch_Pef].[PEF_FK] WHERE
(([qrySch_Pef].[PEF_FK] Is Null) ORDER BY [PEF].[PEF_PK];"
        Me.ListUnselectedPEF.RowSource = strSQL

        ' Update the Selected PEF List
        strSQL = "SELECT [PEF_FK] FROM qrySch_Pef;"
        Me.ListSelectedPEF.RowSource = strSQL

    End Sub

    Private Sub LeftArrow_Click()
        Dim frm As Form, ctl As Control
        Dim varItm As Variant, intI As Integer
        Dim strSQL As String

        ' Ensures the current record is save to the SCHOOL table
        DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, ,

        ' This code enters the selected Target MOS's into the SCH_TGT_MOS
        Set frm = Forms!frmSchoolToPEF_Maint
        Set ctl = frm!ListUnselectedPEF
        For Each varItm In ctl.ItemsSelected
            For intI = 0 To ctl.ColumnCount - 1

                ' This puts the value found in the list box into a text box,
making it readable by the query
                Forms!frmSchoolToPEF_Maint!PEFValue.Value = ctl.Column(intI,

                ' Disables the action query confirmation message
                Application.SetOption "Confirm Action Queries", False

                ' Enters the value into the SCH_TGT_MOS table
                DoCmd.OpenQuery "qryUpdateSCH_PEF"

                ' Enables the action query confirmation message
                Application.SetOption "Confirm Action Queries", True

            Next intI
        Next varItm

        ' Update the Potential PEF list
        strSQL = "SELECT DISTINCTROW [PEF].[PEF_PK] FROM PEF LEFT JOIN
qrySch_Pef ON [PEF].[PEF_PK] = [qrySch_Pef].[PEF_FK] WHERE
(([qrySch_Pef].[PEF_FK] Is Null) ORDER BY [PEF].[PEF_PK];"
        Me.ListUnselectedPEF.RowSource = strSQL

        ' Update the Selected PEF List
        strSQL = "SELECT [PEF_FK] FROM qrySch_Pef;"
        Me.ListSelectedPEF.RowSource = strSQL

    End Sub

    Private Sub RightArrow_Click()
        Dim frm As Form, ctl As Control
        Dim varItm As Variant, intI As Integer
        Dim strSQL As String

        ' This code Delets the selected Target MOS's and its associated
entries from the SCH_TGT_MOS table
        Set frm = Forms!frmSchoolToPEF_Maint
        Set ctl = frm!ListSelectedPEF
        For Each varItm In ctl.ItemsSelected
            For intI = 0 To ctl.ColumnCount - 1

                ' This puts the value found in the list box into a text box,
making it readable by the query
                Forms!frmSchoolToPEF_Maint!PEFValue.Value = ctl.Column(intI,

                ' Disables the action query confirmation message
                Application.SetOption "Confirm Action Queries", False

                ' Deletes the record associated with the specified value from
 the SCH_TGT_MOS table
                DoCmd.OpenQuery "qryDeleteSCH_PEF"

                ' Enables the action query confirmation message
                Application.SetOption "Confirm Action Queries", True

            Next intI
        Next varItm

        ' Update the Potential PEF list
        strSQL = "SELECT DISTINCTROW [PEF].[PEF_PK] FROM PEF LEFT JOIN
qrySch_Pef ON [PEF].[PEF_PK] = [qrySch_Pef].[PEF_FK] WHERE
(([qrySch_Pef].[PEF_FK] Is Null) ORDER BY [PEF].[PEF_PK];"
        Me.ListUnselectedPEF.RowSource = strSQL

        ' Update the Selected PEF List
        strSQL = "SELECT [PEF_FK] FROM qrySch_Pef;"
        Me.ListSelectedPEF.RowSource = strSQL

    End Sub

Form: frmSchoolToPropertyMaint
    Code
    Attribute VB_Name = "Form_frmSchoolToPropertyMaint"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub btnClose_Click()
    On Error GoTo Err_btnClose_Click


        DoCmd.Close .
        DoCmd.OpenForm "frmMaintenanceSwitchboard"

    Exit_btnClose_Click:
        Exit Sub

    Err_btnClose_Click:
        MsgBox Err.Description
        Resume Exit_btnClose_Click

    End Sub


    Private Sub btnDelete_Click()
    On Error GoTo Err_btnDelete_Click


        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_btnDelete_Click:
        Exit Sub

    Err_btnDelete_Click:
        MsgBox Err.Description
        Resume Exit_btnDelete_Click

    End Sub

    Private Sub btnLevel0_Click()
    On Error GoTo Err_btnLevel0_Click

        Dim lngRecordNum As Long
        Dim strSQL As String

        lngRecordNum = Me.CurrentRecord

        DoCmd.Close

        DoCmd.OpenForm "frmSchoolToPropLevelMaint"
        DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
        Forms!frmSchoolToPropLevelMaint!ChosenLevel = 0

        ' Update the selected property list
        strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = "' &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = "' & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = "' & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= "' & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = "' & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'"
        Forms!frmSchoolToPropLevelMaint!listLevel0.RowSource = strSQL


    Exit_btnLevel0_Click:
        Exit Sub

    Err_btnLevel0_Click:
        MsgBox Err.Description
        Resume Exit_btnLevel0_Click

    End Sub
```

```
Private Sub btnLevel1_Click()
On Error GoTo Err_btnLevel1_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 1

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & '"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel1_Click:
    Exit Sub

Err_btnLevel1_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel1_Click

End Sub

Private Sub btnLevel2_Click()
On Error GoTo Err_btnLevel2_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 2

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & '"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel2_Click:
    Exit Sub

Err_btnLevel2_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel2_Click

End Sub

Private Sub btnLevel3_Click()
On Error GoTo Err_btnLevel3_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 3

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & '"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel3_Click:
    Exit Sub

Err_btnLevel3_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel3_Click

End Sub

Private Sub btnLevel4_Click()
On Error GoTo Err_btnLevel4_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 4

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & '"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel4_Click:
    Exit Sub

Err_btnLevel4_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel4_Click

End Sub

Private Sub btnLevel5_Click()
On Error GoTo Err_btnLevel5_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 5

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & '"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel5_Click:
    Exit Sub

Err_btnLevel5_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel5_Click

End Sub

Private Sub btnLevel6_Click()
On Error GoTo Err_btnLevel6_Click

    Dim lngRecordNum As Long
    Dim strSQL As String

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropLevelMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropLevelMaint", acGoTo,
lngRecordNum
    Forms!frmSchoolToPropLevelMaint!ChosenLevel = 6

    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" &
Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & " And
AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" & "
And Level = " & Forms!frmSchoolToPropLevelMaint!ChosenLevel & "'" & "
UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK
= '" & Forms!frmSchoolToPropLevelMaint!SCourseNumber_PK.Value & "'" & "
And AMOS_FK = '" & Forms!frmSchoolToPropLevelMaint!AMOS_PK.Value & "'" &
 And Level =  & Forms!frmSchoolToPropLevelMaint!ChosenLevel & '"
    Forms!frmSchoolToPropLevelMaint!lstLevel0.RowSource = strSQL

Exit_btnLevel6_Click:
    Exit Sub

Err_btnLevel6_Click:
    MsgBox Err.Description
    Resume Exit_btnLevel6_Click

End Sub


Private Sub cmbCourseNumberFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbCourseNumberFind] = Null
    Me.SCourseNumber_PK.SetFocus

End Sub

Private Sub Form_Current()
    Dim strSQL, strInput1, strInput2 As String
    Dim intLevel As Integer

    ' Update the Level 0 list
    intLevel = 0
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "'" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
 Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "'" & " And Level = " & intLevel & '"
    Me.lstLevel0.RowSource = strSQL

    ' Update the Level 1 list
    intLevel = 1
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "'" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
 Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "'" & " And Level = " & intLevel & '"
    Me.lstLevel1.RowSource = strSQL

    ' Update the Level 2 list
    intLevel = 2
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "'" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
 Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
```

```
& "'" & " And Level = " & intLevel & ""
    Me.lstLevel2.RowSource = strSQL

    intLevel = 3
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "'" & " And Level = " & intLevel & ""
    Me.lstLevel3.RowSource = strSQL

    intLevel = 4
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "'" & " And Level = " & intLevel & ""
    Me.lstLevel4.RowSource = strSQL

    intLevel = 5
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "'" & " And Level = " & intLevel & ""
    Me.lstLevel5.RowSource = strSQL

    intLevel = 6
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & intLevel & "" & " UNION
SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE SCourseNumber_FK = '" &
Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK = '" & Me.AMOS_PK.Value
& "'" & " And Level = " & intLevel & ""
    Me.lstLevel6.RowSource = strSQL


End Sub

Form: frmSchoolToPropLevelMaint
  Code
Attribute VB_Name = "Form_frmSchoolToPropLevelMaint"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub btnClose_Click()
On Error GoTo Err_btnClose_Click

    Dim lngRecordNum As Long

    lngRecordNum = Me.CurrentRecord

    DoCmd.Close

    DoCmd.OpenForm "frmSchoolToPropertyMaint"
    DoCmd.GoToRecord acDataForm, "frmSchoolToPropertyMaint", acGoTo,
lngRecordNum

Exit_btnClose_Click:
    Exit Sub

Err_btnClose_Click:
    MsgBox Err.Description
    Resume Exit_btnClose_Click

End Sub

Private Sub btnDelete_Click()
On Error GoTo Err_btnDelete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_btnDelete_Click:
    Exit Sub

Err_btnDelete_Click:
    MsgBox Err.Description
    Resume Exit_btnDelete_Click

End Sub

Private Sub ChosenLevel_AfterUpdate()

    Dim strSQL As String

    ' Update the potential property list
    strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
 FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
[qryUnionAllProperty].[FPropertyName_PK] =
[qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
(([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null));"
    Me.lstLevelUnused.RowSource = strSQL
    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
 & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &

    Me.lstLevel0.RowSource = strSQL

End Sub

Private Sub cmbCourseNumberFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SCourseNumber_PK] = " & Chr(34) &
Me![cmbCourseNumberFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbCourseNumberFind] = Null
    Me.SCourseNumber_PK.SetFocus

End Sub

Private Sub Form_Current()

    Dim strSQL As String

    ' Update the potential property list
    strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
 FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
[qryUnionAllProperty].[FPropertyName_PK] =
[qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
(([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null));"
    Me.lstLevelUnused.RowSource = strSQL
    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
 & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &

    Me.lstLevel0.RowSource = strSQL

End Sub


Private Sub LeftArrow_Click()
    Dim frm As Form, ctl As Control
    Dim varItm As Variant, intI As Integer
    Dim strSQL As String

    ' Ensures the current record is save to the SCHOOL table
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, ,

    ' This code enters the selected Target MOS's into the SCH_TGT_MOS
    Set frm = Forms!frmSchoolToPropLevelMaint
    Set ctl = frm!lstLevelUnused
    For Each varItm In ctl.ItemsSelected
        For intI = 0 To ctl.ColumnCount - 1

            ' This puts the value found in the list box into a text box,
making it readable by the query
            Forms!frmSchoolToPropLevelMaint!TargetValue.Value =
ctl.Column(intI, varItm)

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False
            DoCmd.SetWarnings (False)

            ' Enters the value into the SCH_TGT_MOS table
            DoCmd.OpenQuery "qryUpdateFUND_SCH_PROP"
            DoCmd.OpenQuery "qryUpdateLOG_SCH_PROP"

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True
            DoCmd.SetWarnings (True)

        Next intI
    Next varItm

    ' Update the potential property list
    strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
 FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
[qryUnionAllProperty].[FPropertyName_PK] =
[qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
(([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null));"
    Me.lstLevelUnused.RowSource = strSQL
    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
 & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &

    Me.lstLevel0.RowSource = strSQL

End Sub

Private Sub RightArrow_Click()
    Dim frm As Form, ctl As Control
    Dim varItm As Variant, intI As Integer
    Dim strSQL As String

    ' This code Delets the selected Target MOS's and its associated
entries from the SCH_TGT_MOS table
    Set frm = Forms!frmSchoolToPropLevelMaint
    Set ctl = frm!lstLevel0
    For Each varItm In ctl.ItemsSelected
        For intI = 0 To ctl.ColumnCount - 1

            ' This puts the value found in the list box into a text box,
making it readable by the query
            Forms!frmSchoolToPropLevelMaint!TargetValue.Value =
ctl.Column(intI, varItm)

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False
            DoCmd.SetWarnings (False)

            ' Deletes the record associated with the specified value from
 the FUND or LOG SCH_PROP table
            DoCmd.OpenQuery "qryDeleteFUND_SCH_PROP"
            DoCmd.OpenQuery "qryDeleteLOG_SCH_PROP"

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True
            DoCmd.SetWarnings (True)

        Next intI
    Next varItm

    ' Update the potential property list
    strSQL = "SELECT DISTINCTROW [qryUnionAllProperty].[FPropertyName_PK]
 FROM qryUnionAllProperty LEFT JOIN qryUnionPropertyByLevel ON
[qryUnionAllProperty].[FPropertyName_PK] =
[qryUnionPropertyByLevel].[FPropertyName_FK] WHERE
(([qryUnionPropertyByLevel].[FPropertyName_FK] Is Null));"
    Me.lstLevelUnused.RowSource = strSQL
    ' Update the selected property list
    strSQL = "SELECT [FPropertyName_FK] FROM FUND_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &
 & " UNION SELECT [LPropertyName_FK] FROM LOG_SCH_PROP WHERE
SCourseNumber_FK = '" & Me.SCourseNumber_PK.Value & "'" & " And AMOS_FK =
'" & Me.AMOS_PK.Value & "'" & " And Level = " & Me.ChosenLevel.Value &

    Me.lstLevel0.RowSource = strSQL

End Sub

Form: frmScrubMarine
  Code
Attribute VB_Name = "Form_frmScrubMarine"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
```

```
    Option Explicit

    Private Sub Close_Click()
    On Error GoTo Err_Close_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close current form
        DoCmd.Close

        ' Open specified form
        stDocName = "frmPreprocessing&ExecutionSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Close_Click:
        Exit Sub

    Err_Close_Click:
        MsgBox Err.Description
        Resume Exit_Close_Click

    End Sub

    Private Sub cmbPefFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[PEF] = " & Chr(34) & Me![cmbPefFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbPefFind] = Null
        Me.SSN_PK.SetFocus

    End Sub

    Private Sub cmbSSNFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[SSN_PK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbSSNFind] = Null
        Me.Close.SetFocus

    End Sub

    Private Sub Delete_Click()
    On Error GoTo Err_Delete_Click


        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_Delete_Click:
        Exit Sub

    Err_Delete_Click:
        MsgBox Err.Description
        Resume Exit_Delete_Click

    End Sub

    Private Sub Form_Current()
        Dim strSQL As String

        ' Update the PEF List
        strSQL = "SELECT DISTINCTROW [MARINE].[PEF]FROM MARINE LEFT JOIN PEF
    ON [MARINE].[PEF] = [PEF].[PEF_PK]WHERE ([PEF].[PEF_PK] Is Null)ORDER BY
        Me.cmbPefFind.RowSource = strSQL

    End Sub

Form: frmSpecialAssignment
    Code
    Attribute VB_Name = "Form_frmSpecialAssignment"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub cmbClassNumber_AfterUpdate()

        Dim strConvert As String

        Me.txtReportDate = Me.cmbClassNumber.Column(2)
        Me.txtMCC = Me.cmbClassNumber.Column(4)

        ' Convert the fiscal year to a two digit number for RD3 File
        strConvert = CStr(Me.cmbClassNumber.Column(3))
        strConvert = Right(strConvert, 2)
        Me.txtFY = CInt(strConvert)

        Me.txtAssignmentType = "S"

    End Sub

    Private Sub cmbClassNumber_Enter()

        Dim strSQL As String

        ' This query finds the class numbers, report dates and class
    convening dates associated with the chosen school
        strSQL = "SELECT BNA_EXTRACT.ClassNumber_PK,
    BNA_EXTRACT.ClassConvDate, BNA_EXTRACT.ReportDate,
    BNA_EXTRACT.FiscalYear_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT INNER JOIN
    SCH_TGT_MOS ON (BNA_EXTRACT.CourseNumber_PK =
    SCH_TGT_MOS.TCourseNumber_FK) AND (BNA_EXTRACT.TargetMOS_PK =
    SCH_TGT_MOS.TargetMOS_FK) WHERE BNA_EXTRACT.CourseNumber_PK =
    Forms!frmSpecialAssignment!cmbCourseNumber AND SCH_TGT_MOS.AMOS_FK =
    Forms!frmSpecialAssignment!txtAMOS;"
        Me.cmbClassNumber.RowSource = strSQL

    End Sub

    Private Sub cmbCourseNumber_AfterUpdate()

        Me.txtAMOS = Me!cmbCourseNumber.Column(1)

    End Sub

    Private Sub cmbSSN_AfterUpdate()

        Me.txtSOI = Me!cmbSSN.Column(1)
        Me.txtGradDate = Me!cmbSSN.Column(2)

    End Sub

    Private Sub cmbSSN_Enter()

        Dim strSQL As String

        strSQL = "SELECT DISTINCTROW MARINE.SSN_PK, MARINE.SOI,
    MARINE.GradDate FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK =
    ASSIGNMENT.SSN_FK WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
        Me.cmbSSN.RowSource = strSQL

    End Sub

    Private Sub cmbSSNFind_AfterUpdate()

        Dim R As Recordset
        Set R = Me.RecordsetClone
        R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
        Me.Bookmark = R.Bookmark
        Me![cmbSSNFind] = Null
        Me.Close.SetFocus

    End Sub

    Private Sub Delete_Click()
    On Error GoTo Err_Delete_Click


        DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
        DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

    Exit_Delete_Click:
        Exit Sub

    Err_Delete_Click:
        MsgBox Err.Description
        Resume Exit_Delete_Click

    End Sub

    Private Sub Close_Click()
    On Error GoTo Err_Close_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        ' Close current form
        DoCmd.Close

        ' Open specified form
        stDocName = "frmPreprocessing&ExecutionSwitchboard"
        DoCmd.OpenForm stDocName, , , stLinkCriteria

    Exit_Close_Click:
        Exit Sub

    Err_Close_Click:
        MsgBox Err.Description
        Resume Exit_Close_Click

    End Sub

    Private Sub Form_Current()

        ' Prevents user from accidentially making special assignments to
    marines assigned normally
        If IsNull(Me.cmbSSN.Value) = False Then
            Me!cmbSSN.Enabled = False
            Me!cmbCourseNumber.Enabled = False
            Me!cmbClassNumber.Enabled = False

        ' Allows the user to enter values for the fundamental equation
        Else
            Me!cmbSSN.Enabled = True
            Me!cmbCourseNumber.Enabled = True
            Me!cmbClassNumber.Enabled = True

        End If

    End Sub

    Private Sub Form_Open(Cancel As Integer)


        DoCmd.GoToRecord acDataForm, "frmSpecialAssignment", acNewRec


    End Sub

Form: frmSpecialAssignmentOfUnassignedMarine
    Code
    Attribute VB_Name = "Form_frmSpecialAssignmentOfUnassignedMarine"
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = True
    Attribute VB_Exposed = False
    Option Compare Database
    Option Explicit

    Private Sub cmbClassNumber_AfterUpdate()

        Dim strConvert As String
        Dim strSSN As String
        Dim R As Recordset

        Me.txtReportDate = Me.cmbClassNumber.Column(2)
        Me.txtMCC = Me.cmbClassNumber.Column(4)

        ' Convert the fiscal year to a two digit number for RD3 File
        strConvert = CStr(Me.cmbClassNumber.Column(3))
        strConvert = Right(strConvert, 2)
        Me.txtFY = CInt(strConvert)

        Me.txtAssignmentType = "S"

        ' Keep track of the current SSN
        strSSN = Me.cmbSSN

        Me.txtAMOS = Me!cmbCourseNumber.Column(1)

        ' Updates the current form and subform
        Me.Requery
        Me.Repaint

        ' Moves the record back to the current SSN
        Set R = Me.RecordsetClone
        R.FindFirst "[SSN_FK] = " & Chr(34) & strSSN & Chr(34)
        Me.Bookmark = R.Bookmark

    End Sub

    Private Sub cmbClassNumber_Enter()

        Dim strSQL As String

    ' This query finds the class numbers, report dates and class
convening dates associated with the chosen school
    strSQL = "SELECT BNA_EXTRACT.ClassNumber_PK,
BNA_EXTRACT.ClassConvDate, BNA_EXTRACT.ReportDate,
BNA_EXTRACT.FiscalYear_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT INNER JOIN
SCH_TGT_MOS ON (BNA_EXTRACT.CourseNumber_PK =
SCH_TGT_MOS.TCourseNumber_FK) AND (BNA_EXTRACT.TargetMOS_PK =
SCH_TGT_MOS.TargetMOS_FK) WHERE BNA_EXTRACT.CourseNumber_PK =
Forms!frmSpecialAssignmentOfUnassignedMarine!cmbCourseNumber AND
SCH_TGT_MOS.AMOS_FK =
    Me.cmbClassNumber.RowSource = strSQL

End Sub

Private Sub cmbCourseNumber_AfterUpdate()

    Dim strSSN As String
    Dim R As Recordset

    strSSN = Me.cmbSSN

    ' Keep track of the current SSN
    strSSN = Me.cmbSSN

    Me.txtAMOS = Me!cmbCourseNumber.Column(1)

    ' Updates the current form and subform
    Me.Requery
    Me.Repaint

    ' Moves the record back to the current SSN
    Set R = Me.RecordsetClone
    R.FindFirst "[SSN_FK] = " & Chr(34) & strSSN & Chr(34)
    Me.Bookmark = R.Bookmark

End Sub

Private Sub cmbSSN_AfterUpdate()

    Me.txtSOI = Me!cmbSSN.Column(1)
    Me.txtGradDate = Me!cmbSSN.Column(2)

End Sub

Private Sub cmbSSN_Enter()

    Dim strSQL As String

    strSQL = "SELECT DISTINCTROW MARINE.SSN_PK, MARINE.SOI,
MARINE.GradDate FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK =
ASSIGNMENT.SSN_FK WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
    Me.cmbSSN.RowSource = strSQL

End Sub

Private Sub cmbSSNFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SSN_FK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbSSNFind] = Null
    Me.Close.SetFocus

End Sub

Private Sub Delete_Click()
On Error GoTo Err_Delete_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_Delete_Click:
    Exit Sub

Err_Delete_Click:
    MsgBox Err.Description
    Resume Exit_Delete_Click

End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click

    ' Close current form
    DoCmd.Close

Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub

Private Sub Form_Current()

        Me!cmbSSN.Enabled = True
        Me!cmbCourseNumber.Enabled = True
        Me!cmbClassNumber.Enabled = True



End Sub

Private Sub Form_Open(Cancel As Integer)


    DoCmd.GoToRecord acDataForm,
frmSpecialAssignmentOfUnassignedMarine, acNewRec


End Sub

Form: frmUnassignedMarines
    Code
Attribute VB_Name = "Form_frmUnassignedMarines"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub Close_Click()
On Error GoTo Err_Close_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    ' Close the current form
    DoCmd.Close
    ' Open specified form
    stDocName = "frmAnalyzeResult"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Close_Click:
    Exit Sub

Err_Close_Click:
    MsgBox Err.Description
    Resume Exit_Close_Click

End Sub

Private Sub cmbSSNFind_AfterUpdate()

    Dim R As Recordset
    Set R = Me.RecordsetClone
    R.FindFirst "[SSN_PK] = " & Chr(34) & Me![cmbSSNFind] & Chr(34)
    Me.Bookmark = R.Bookmark
    Me![cmbSSNFind] = Null
    Me.Close.SetFocus

End Sub

Private Sub Form_Current()

    Dim strSQL As String
    Dim rec As Recordset
    Dim db1 As Database

    Set db1 = CurrentDb()

    ' Calculates the number of unassigned marines.
    strSQL = "SELECT Count(PEF) AS TotalUnassigned FROM
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    Me.txtTotalUnassigned = rec!TotalUnassigned

End Sub
Private Sub btnAssign_Click()
On Error GoTo Err_btnAssign_Click

    Dim stDocName As String

    stDocName = "frmSpecialAssignmentOfUnassignedMarine"

    DoCmd.OpenForm stDocName

Exit_btnAssign_Click:
    Exit Sub

Err_btnAssign_Click:
    MsgBox Err.Description
    Resume Exit_btnAssign_Click

End Sub

Form: subfrmFundementalProperty
    Code
Attribute VB_Name = "Form_subfrmFundementalProperty"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit
Private Sub cmbValue_FK_AfterUpdate()

    Forms!frmFundementalProperty![FPropertyName_PK].SetFocus

End Sub

Private Sub cmbValue_FK_BeforeUpdate(Cancel As Integer)

    On Error GoTo Err_cmbValue_FK_BeforeUpdate

        Dim db As Database
        Dim rec As Recordset
        Dim strSQL As String
        Dim Convert As Variant

        ' Places value in a form agreeable to Access (e.g. 'HT' or '10')
        Convert = "'" &
Forms!frmFundementalProperty!subfrmFundementalProperty!cmbValue_FK & "'"

        ' Opens the table PROPERTY_VALUE
        strSQL = "PROPERTY_VALUE"
        Set db = CurrentDb()
        Set rec = db.OpenRecordset(strSQL, dbOpenSnapshot)

        ' Searches for the value entered by the user in the
        rec.FindFirst "Value_PK = " & Convert

        ' If the value is not in the PROPERTY_VALUE table, it is added.
        If rec.NoMatch = True Then

            ' Disables the action query confirmation message
            Application.SetOption "Confirm Action Queries", False

            ' Enters the value into the PROPERTY_VALUE table
            DoCmd.OpenQuery "qryUpdateProperty_Value"

            ' Updates the drop down list for the combobox
            strSQL = "SELECT DISTINCT " &
Forms!frmFundementalProperty!MarineField & " FROM MARINE;"

Forms!frmFundementalProperty!subfrmFundementalProperty!cmbValue_FK.RowSou
rce = strSQL

            ' Enables the action query confirmation message
            Application.SetOption "Confirm Action Queries", True
        End If

        rec.Close

Exit_cmbValue_FK_BeforeUpdate:
    Exit Sub

Err_cmbValue_FK_BeforeUpdate:
    MsgBox Err.Description
    Application.SetOption "Confirm Action Queries", True
    Resume Exit_cmbValue_FK_BeforeUpdate

End Sub


Form: subfrmFundementalPropertyList
    Code
Attribute VB_Name = "Form_subfrmFundementalPropertyList"
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

Private Sub cmbValue_FK_BeforeUpdate(Cancel As Integer)
  On Error GoTo Err_cmbValue_FK_BeforeUpdate

    Dim db As Database
    Dim rec As Recordset
    Dim strSQL As String
    Dim Convert As Variant

    'Opens the subfrmFundamentalProperty so the following code will
    DoCmd.OpenForm "subfrmFundamentalProperty", , , , , acHidden


    ' Places value in a form agreeable to Access (e.g. 'HT' or '10')
    Convert = "'" & Forms![subfrmFundamentalProperty]!cmbValue_FK &
    ' Opens the table PROPERTY_VALUE
    strSQL = "PROPERTY_VALUE"

    Set db = CurrentDb()
    Set rec = db.OpenRecordset(strSQL, dbOpenSnapshot)

    rec.FindFirst "Value_PK = " & Convert
    If rec.NoMatch = True Then
      ' Disables the action query confirmation message
      Application.SetOption "Confirm Action Queries", False

      DoCmd.OpenQuery "qryUpdateProperty_Value"
      strSQL = "SELECT DISTINCT " &
Forms!frmFundamentalProperty!MarineField & " FROM MARINE;"
      Me.cmbValue_FK.RowSource = strSQL

      ' Enables the action query confirmation message
      Application.SetOption "Confirm Action Queries", True
    End If

    rec.Close

    'Closes the subfrmFundamentalProperty without saving changes to
    DoCmd.Close acForm, "subfrmFundamentalProperty"


Exit_cmbValue_FK_BeforeUpdate:
  Exit Sub

Err_cmbValue_FK_BeforeUpdate:
  MsgBox Err.Description
  Application.SetOption "Confirm Action Queries", True
  Resume Exit_cmbValue_FK_BeforeUpdate

End Sub

Private Sub Value_FK_BeforeUpdate(Cancel As Integer)
  On Error GoTo Err_Value_FK_BeforeUpdate

    Dim db As Database
    Dim rec As Recordset
    Dim strSQL As String
    Dim Convert As Variant

    ' Places value in a form agreeable to Access (e.g. 'HT' or '10')
    Convert = "'" &
Forms!frmFundamentalProperty!subfrmFundamentalPropertyList!Value_FK & "'"
    ' Opens the table PROPERTY_VALUE
    strSQL = "PROPERTY_VALUE"

    Set db = CurrentDb()
    Set rec = db.OpenRecordset(strSQL, dbOpenSnapshot)

    rec.FindFirst "Value_PK = " & Convert
    ' This If Then statement checks to see if the entered value is in
    ' PROPERTY_VALUE table. If not, it is added to the table.
    If rec.NoMatch = True Then
      ' Disables the action query confirmation message
      Application.SetOption "Confirm Action Queries", False

      DoCmd.OpenQuery "qryUpdateProperty_ValueList"

      ' Enables the action query confirmation message
      Application.SetOption "Confirm Action Queries", True
    End If

    rec.Close

Exit_Value_FK_BeforeUpdate:
  Exit Sub

Err_Value_FK_BeforeUpdate:
  MsgBox Err.Description
  Application.SetOption "Confirm Action Queries", True
  Resume Exit_Value_FK_BeforeUpdate

End Sub


Module: modFitnessDetermination
  Code
Attribute VB_Name = "modFitnessDetermination"
Option Compare Database
Option Explicit

Dim lngStart, lngEnd As Long


Public Function FundPropTest(FundPropName As String, SSN As String) As

    Dim rec As Recordset, rec2 As Recordset, rec3 As Recordset
    Dim strMarField As String, strFundOperator As String, strSQL As
    Dim strMarValueSQL As String, strFundValueSQL As String, strInput As
    Dim varFundValue, varMarValue
    Dim db1 As Database

    FundPropName = "'" & FundPropName & "'"      'This expression
properly formates the argument FundPropName, by placing a ' in front of
and behind the value found in FundPropName.

    Set db1 = CurrentDb()

    strSQL = "SELECT MarineField, Operator FROM FUNDAMENTAL_PROPERTY
WHERE FPropertyName_PK = " & FundPropName
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    strMarField = rec!MarineField

    strFundOperator = rec!Operator


    strInput = "'" & SSN & "'"
    strMarValueSQL = "SELECT " & strMarField & " AS [Value] FROM MARINE
WHERE SSN_PK = " & strInput
    Set rec2 = db1.OpenRecordset(strMarValueSQL, dbOpenSnapshot)
    varMarValue = rec2!Value

    strFundValueSQL = "SELECT Value_FK FROM FUND_PROP_VAL WHERE
FPropertyName_FK = " & FundPropName
    Set rec3 = db1.OpenRecordset(strFundValueSQL, dbOpenSnapshot)
    varFundValue = rec3!Value_FK

    ' Convert varFundValue and varMarValue to a numeric, if possible
    If IsNumeric(varFundValue) And IsNumeric(varMarValue) Then
      varFundValue = CInt(varFundValue)
      varMarValue = CInt(varMarValue)
    Else
      varFundValue = CStr(varFundValue)
      varMarValue = CStr(varMarValue)
    End If




    Select Case strFundOperator
      Case "="
        If varMarValue = varFundValue Then
          FundPropTest = True
        End If
      Case "not ="
        If varMarValue <> varFundValue Then
          FundPropTest = True
        End If
      Case "<"
        If varMarValue < varFundValue Then
          FundPropTest = True
        End If
      Case ">"
        If varMarValue > varFundValue Then
          FundPropTest = True
        End If
      Case "<="
        If varMarValue <= varFundValue Then
          FundPropTest = True
        End If
      Case ">="
        If varMarValue >= varFundValue Then
          FundPropTest = True
        End If
      Case "in"
        Do Until rec3.EOF
          varFundValue = rec3!Value_FK
          If IsNumeric(varFundValue) And IsNumeric(varMarValue)
            varFundValue = CInt(varFundValue)
            varMarValue = CInt(varMarValue)
          Else
            varFundValue = CStr(varFundValue)
            varMarValue = CStr(varMarValue)
          End If
          If varMarValue = varFundValue Then
            FundPropTest = True
          End If
          rec3.MoveNext
        Loop
    End Select
    rec2.Close
    rec3.Close
    db1.Close

End Function
Public Function LogOrTest(Input1 As Boolean, Input2 As Boolean) As

If Input1 Or Input2 Then
    LogOrTest = True


Else: LogOrTest = False
End If

End Function
Public Function LogAndTest(Input1 As Boolean, Input2 As Boolean) As

If Input1 And Input2 Then
    LogAndTest = True
Else: LogAndTest = False
End If

End Function
Public Function LogPropTest(LogPropName As String, SSN As String) As
    Dim intNewUBound As Integer, j As Integer, k As Integer, l As
Integer, m As Integer
    Dim booOrOperator As Boolean, booAndOperator As Boolean,
booNotOperator As Boolean
    Dim strLogicalEquation As String, strSQL As String, strInput As
    Dim strWorkingEqn() As String
    Dim intSpacePos As Integer      ' position of the space
    Dim db1 As Database
    Dim rec As Recordset

    Set db1 = CurrentDb()

    ' Get the logical equation from the LOGICAL_PROPERTY table
    strInput = "'" & LogPropName & "'"
    strSQL = "SELECT LogicalEquation FROM LOGICAL_PROPERTY WHERE
LPropertyName_PK = " & strInput & ""
    Set rec = db1.OpenRecordset(strSQL, dbOpenSnapshot)
    strLogicalEquation = rec!LogicalEquation

    ' Parse the string into an array
    ' declare the temporary string as an array
    ReDim strWorkingEqn(0) As String

    ' Find the position of the first space
    intSpacePos = InStr(strLogicalEquation, " ")
    ' loop until no more spaces are found
    ' InStr returns 0 if the space is not found
    Do Until intSpacePos = 0
      ' redimension the array, adding another element
      ' UBound tells us how many elements there already are in the
      ReDim Preserve strWorkingEqn(UBound(strWorkingEqn) + 1)

      ' set the new element to the characters from the beginning of the
      ' string up to the first space.  Left$ returns a number of
      ' from the beginning of a string
      strWorkingEqn(UBound(strWorkingEqn)) =
Trim$(Left$(strLogicalEquation, intSpacePos - 1))

      ' now we've copied these characters into the name, they can be


      ' removed from the original string.  Right$ returns a number of
      ' characters from the end of the string.  It is the rightmost N
```

```
        ' characters, therefore we subtract the position of the space
        ' the length of the string.  This gives us the number of characters remaining.
        strLogicalEquation = LTrim$(Right$(strLogicalEquation, Len(strLogicalEquation) - intSpacePos))

        ' find the next space
        intSpacePos = InStr(strLogicalEquation, " ")
    Loop

    For j = 1 To (UBound(strWorkingEqn))
        If strWorkingEqn(j) = "(" Then

Line1:
            For k = (j + 1) To (UBound(strWorkingEqn))
                Select Case strWorkingEqn(k)
                    Case ")"
                        For l = (j + 1) To (k - 1)
                            Select Case strWorkingEqn(l)
                                Case "Or"
                                    booOrOperator = True
                                Case "And"
                                    booAndOperator = True
                                Case "Not"
                                    booNotOperator = True
                                Case "True"
                                    If strWorkingEqn(j) = "(" Then
                                        If booNotOperator Then
                                            strWorkingEqn(j) = "False"
                                            booNotOperator = False
                                        Else
                                            strWorkingEqn(j) = "True"
                                        End If
                                    Else    ' Executed after the first bracketed property is tested and stored in the location of "("
                                        If booNotOperator Then
                                            strWorkingEqn(l) = "False"
                                            booNotOperator = False
                                        End If
                                        If booOrOperator Then
                                            strWorkingEqn(j) = LogOrTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(l)))
                                            booOrOperator = False
                                        ElseIf booAndOperator Then
                                            strWorkingEqn(j) = LogAndTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(l)))
                                            booAndOperator = False
                                        End If
                                    End If
                                Case "False"
                                    If strWorkingEqn(j) = "(" Then
                                        If booNotOperator Then
                                            strWorkingEqn(j) = "True"
                                            booNotOperator = False
                                        Else
                                            strWorkingEqn(j) = "False"
                                        End If
                                    Else    ' Executed after the first bracketed property is tested and stored in the location of "("
                                        If booNotOperator Then
                                            strWorkingEqn(l) = "True"
                                            booNotOperator = False
                                        End If
                                        If booOrOperator Then
                                            strWorkingEqn(j) = LogOrTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(l)))
                                            booOrOperator = False
                                        ElseIf booAndOperator Then
                                            strWorkingEqn(j) = LogAndTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(l)))
                                            booAndOperator = False
                                        End If
                                    End If
                                Case ")"    'Do nothing
                                Case Else
                                    If strWorkingEqn(j) = "(" Then
                                        If booNotOperator Then
                                            strWorkingEqn(j) = Not (FundPropTest(strWorkingEqn(l), SSN))
                                            booNotOperator = False
                                        Else
                                            strWorkingEqn(j) = FundPropTest(strWorkingEqn(l), SSN)
                                        End If
                                    Else    ' Executed after the first bracketed property is tested and stored in the location of "("
                                        If booNotOperator Then
                                            strWorkingEqn(l) = Not (FundPropTest(strWorkingEqn(l), SSN))
                                            booNotOperator = False
                                        Else
                                            strWorkingEqn(l) = FundPropTest(strWorkingEqn(l), SSN)
                                        End If
                                        If booOrOperator Then
                                            strWorkingEqn(j) = LogOrTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(l)))
                                            booOrOperator = False
                                        ElseIf booAndOperator Then
                                            strWorkingEqn(j) = LogAndTest(CBool(strWorkingEqn(j)), CBool(strWorkingEqn(l)))
                                            booAndOperator = False
                                        End If
                                    End If
                            End Select
                        Next l

                        ' Must transform the working equation to account for the bracketed expression evaluation.

                        intNewUBound = UBound(strWorkingEqn) - (k - j)
                        If intNewUBound <> 1 Then
                            ReDim strTransformedEqn(1 To intNewUBound)
                            For m = 1 To j
                                strTransformedEqn(m) = strWorkingEqn(m)
                            Next m
                            For m = (k + 1) To UBound(strWorkingEqn)
                                strTransformedEqn(m - k + j) =
                            Next m
                            ReDim strWorkingEqn(1 To intNewUBound)
                            For m = 1 To intNewUBound
                                strWorkingEqn(m) = strTransformedEqn(m)
                            Next m
                            j = 1   'Set j back to a value of 1.
                            GoTo Line1    'Jumps back to For j = 1 to (UBound(strWorkingEqn)loop
                        Else
                            LogPropTest = strWorkingEqn(j)
                            Erase strWorkingEqn
                            Erase strTransformedEqn
                            Exit Function
                        End If
                    Case "("
                        j = k
                        GoTo Line1    'Jumps back to For j = 1 to (UBound(strWorkingEqn)loop
                        'Exit For    'Exits the For k = (j + 1) to (UBound(strWorkingEqn)) loop
                End Select
            Next k
        End If
    Next j

End Function

Public Sub Fitness()
    Dim strSQL As String, strClassNumber As String, strAMOS As String
    Dim strInput1 As String, strInput2 As String
    Dim i As Integer, j As Integer, k As Integer, l As Integer
    Dim intNormalFactor As Integer, intFit As Integer, intInput As
    Dim intLevel1 As Integer, intLevel2 As Integer, intLevel3 As Integer
    Dim intLevel4 As Integer, intLevel5 As Integer, intLevel6 As Integer
    Dim sngDesiredWeight As Single, sngEpsilon As Single, sngSumOfFit As
    Dim sngCount As Single, sngAvgFit As Single, sngFitness As Single
    Dim sngMandFitValue As Single
    Dim lngGradDate As Long
    Dim db1 As Database
    Dim recClass As Recordset, recMarine As Recordset, recMarClsFit As
    Dim recPEF As Recordset, recFDesired As Recordset, recLDesired As
    Dim recFMand As Recordset, recLMand As Recordset
    Dim booMandProp As Boolean, booPEF As Boolean, booMandatory As
    Dim varRetVal As Variant
    Dim frmCurrent As Form_frmClassQuotaPenaltyAndFit

    Set frmCurrent = Forms!frmClassQuotaPenaltyAndFit
    Set db1 = CurrentDb()

    ' Create TEMP_FIT table
    Application.SetOption "Confirm Action Queries", False

    lngGradDate = frmCurrent!MCTGradDate


    ' Create 2 TEMP_FIT tables from MAR_CLS_FIT table
    db1.Execute "SELECT SSN_FK, MarineIndex, ClassIndex_FK, Fitness INTO TEMP_FIT FROM MAR_CLS_FIT;"
    db1.Execute "SELECT SSN_FK, MarineIndex, ClassIndex_FK, Fitness INTO TEMP_FIT1 FROM MAR_CLS_FIT;"

    ' Insert into the TEMP_FIT tables the applicable SSNs based on MCT graduation date
    db1.Execute "INSERT INTO TEMP_FIT ( SSN_FK )SELECT MARINE.SSN_PK FROM qrySSN WHERE GradDate = " & lngGradDate & ""
    db1.Execute "INSERT INTO TEMP_FIT1 ( SSN_FK )SELECT MARINE.SSN_PK FROM qrySSN WHERE GradDate = " & lngGradDate & ""

    Application.SetOption "Confirm Action Queries", True

    ' Create a dynaset of the temp_fit table
    strSQL = "SELECT * FROM TEMP_FIT"
    Set recMarClsFit = db1.OpenRecordset(strSQL, dbOpenDynaset)
    recMarClsFit.MoveLast

    ' Create the MarineIndex for the Temp_fit table
    If recMarClsFit.RecordCount > 0 Then
        recMarClsFit.MoveFirst
        For i = 1 To recMarClsFit.RecordCount
            recMarClsFit.Edit
            recMarClsFit!MarineIndex = i
            recMarClsFit.Update
            recMarClsFit.MoveNext
        Next i
    End If

    ' Create a dynaset of the Class table
    strSQL = "SELECT SCourseNumber_FK, AMOS_FK FROM CLASS"
    Set recClass = db1.OpenRecordset(strSQL, dbOpenDynaset)
        If recClass.EOF = False Then
            recClass.MoveLast
            recClass.MoveFirst
        End If

    ' Create a dynaset of the Marine table based on TEMP_FIT1 table
    strSQL = "SELECT * FROM MARINE , TEMP_FIT1 WHERE (((MARINE.SSN_PK)=[TEMP_FIT1].[SSN_FK]));"
    Set recMarine = db1.OpenRecordset(strSQL, dbOpenDynaset)

    If recMarine.EOF = False Then
        recMarine.MoveLast
        recMarine.MoveFirst
    End If

    ' Initialize variables
    booMandProp = True
    intFit = 0
    booPEF = False
    booMandatory = False
    intNormalFactor = 100
    intLevel1 = 32
    intLevel2 = 16
    intLevel3 = 8
    intLevel4 = 4
    intLevel5 = 2
    intLevel6 = 1
    sngDesiredWeight =
    sngMandFitValue = (1 - sngDesiredWeight) * intNormalFactor
    varRetVal = SysCmd(acSysCmdInitMeter, "Status...",
    sngEpsilon = 0.000001

    ' Verify recClass has values
    If recClass.RecordCount > 0 Then
        strClassNumber = "X"
        strAMOS = "X"

        ' Cycle through all the classes
        For i = 1 To recClass.RecordCount
            ' Yeild control temporarily to Windows
            DoEvents
            ' Update status bar
            varRetVal = SysCmd(acSysCmdUpdateMeter, i)

            ' Update the classindex_FK for the MarClsFit record
            If recMarClsFit.RecordCount > 0 Then
                recMarClsFit.MoveFirst
                For j = 1 To recMarClsFit.RecordCount
                    recMarClsFit.Edit
                    recMarClsFit!ClassIndex_FK = i
                    recMarClsFit.Update
                    recMarClsFit.MoveNext
                Next j
            recMarClsFit.MoveFirst
        End If
        ' Check to see if next record is the same school as the last
        ' If it is, append it to the MAR_CLS_FIT table
        If (recClass!SCourseNumber_FK = strClassNumber) And
(recClass!AMOS_FK = strAMOS) Then
            Application.SetOption "Confirm Action Queries", False
            db1.Execute "INSERT INTO MAR_CLS_FIT ( SSN_FK,
MarineIndex, ClassIndex_FK, Fitness ) SELECT TEMP_FIT.SSN_FK,
TEMP_FIT.MarineIndex, TEMP_FIT.ClassIndex_FK, TEMP_FIT.Fitness FROM
TEMP_FIT WHERE TEMP_FIT.Fitness > 0;"
            Application.SetOption "Confirm Action Queries", True
            recClass.MoveNext
        Else
            sngSumOfFit = 0
            sngCount = 0
            booPEF = False
            'Verify recMarine has values
            If recMarine.RecordCount > 0 Then

                ' Initialize variables
                strInput1 = "" & recClass!SCourseNumber_FK & ""
                strInput2 = "" & recClass!AMOS_FK & ""
                intInput = 0

                ' Find all PEF's associated with the current school
                strSQL = "SELECT PEF_FK FROM SCH_PEF WHERE
SCourseNumber_FK = " & strInput1 & " and AMOS_FK = " & strInput2 & ""
                Set recPEF = db1.OpenRecordset(strSQL,
                If recPEF.EOF = False Then
                    recPEF.MoveLast
                    recPEF.MoveFirst
                End If

                ' Find mandatory fundamental properties associated
with the current school
                strSQL = "SELECT FPropertyName_FK FROM FUND_SCH_PROP
WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = " & strInput2 &
 AND Level =  & intInput & ""
                Set recFMand = db1.OpenRecordset(strSQL,
                If recFMand.EOF = False Then
                    recFMand.MoveLast
                    recFMand.MoveFirst
                End If

                ' Find mandatory logical properties associated with
the current school
                strSQL = "SELECT LPropertyName_FK FROM LOG_SCH_PROP
WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = " & strInput2 &
 AND Level =  & intInput & ""
                Set recLMand = db1.OpenRecordset(strSQL,
                If recLMand.EOF = False Then
                    recLMand.MoveLast
                    recLMand.MoveFirst
                End If

                ' Find desired fundamental properties associated with
 the curren school
                strSQL = "SELECT FPropertyName_FK, [Level] FROM
FUND_SCH_PROP WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = "
& strInput2 & " AND Level <> " & intInput & ""
                Set recFDesired = db1.OpenRecordset(strSQL,
                If recFDesired.EOF = False Then
                    recFDesired.MoveLast
                    recFDesired.MoveFirst
                End If

                ' Find desired logical properties associated with the
 curren school
                strSQL = "SELECT LPropertyName_FK, [Level] FROM
LOG_SCH_PROP WHERE SCourseNumber_FK = " & strInput1 & " AND AMOS_FK = " &
 strInput2 & " AND Level <> " & intInput & ""
                Set recLDesired = db1.OpenRecordset(strSQL,
                If recLDesired.EOF = False Then
                    recLDesired.MoveLast
                    recLDesired.MoveFirst
                End If

                ' Cycle through all the marines
                For j = 1 To recMarine.RecordCount
                    ' Yeild control temporarily to Windows
                    DoEvents
                    ' Check to see if marine's PEF is an open
                    If recMarine!PEF <> "00" Then
                        booPEF = False
                        If recPEF.EOF = False Then
                            ' Check to see if marine's PEF is
associated with the school
                            For k = 1 To recPEF.RecordCount
                                If recPEF!PEF_FK = recMarine!PEF Then
                                    booPEF = True
                                End If
                                recPEF.MoveNext
                            Next k
                            recPEF.MoveFirst
                        End If
                    Else
                        booPEF = True
                    End If
                    ' Compare marine attributes with the mandatory
properties
                    If booPEF = True Then
                        ' Check for mandatory fundamental properties
                        If recFMand.EOF = False Then
                            For k = 1 To recFMand.RecordCount
                                ' All tests must prove true for the
mandatory properties
                                If booMandProp = True Then
                                    If
FundPropTest(recFMand!FPropertyName_FK, recMarine!SSN_PK) = False Then
                                        booMandProp = False
                                    End If
                                    recFMand.MoveNext
                                End If
                            Next k
                            recFMand.MoveFirst
                        End If
                        ' Check for mandatory logical properties
                        If booMandProp = True Then
                            If recLMand.EOF = False Then
                                For k = 1 To recLMand.RecordCount
                                    ' All tests must prove true for
the mandatory properties
                                    If booMandProp = True Then
                                        If
LogPropTest(recLMand!LPropertyName_FK, recMarine!SSN_PK) = False Then
                                            booMandProp = False
                                        End If
                                        recLMand.MoveNext
                                    End If
                                Next k
                                recLMand.MoveFirst
                            End If
                            If booMandProp = True Then
                                intFit = 1
                            End If
                        End If
                        ' Reset booMandProp for next check
                        booMandProp = True
                        ' Compare marine attributes with the desired
properties
                        If intFit > 0 Then
                            ' Compare with desired fundamental
                            If recFDesired.EOF = False Then
                                For l = 1 To recFDesired.RecordCount
                                    If
FundPropTest(recFDesired!FPropertyName_FK, recMarine!SSN_PK) = True Then
                                        Select Case recFDesired!Level
                                            Case 1
                                                intFit = intFit +
                                            Case 2
                                                intFit = intFit +
                                            Case 3
                                                intFit = intFit +
                                            Case 4
                                                intFit = intFit +
                                            Case 5
                                                intFit = intFit +
                                            Case 6
                                                intFit = intFit +
                                        End Select
                                    End If
                                    recFDesired.MoveNext
                                Next l
                                recFDesired.MoveFirst
                            End If
                            ' Compare with desired logical properties
                            If recLDesired.EOF = False Then
                                For l = 1 To recLDesired.RecordCount
                                    If
FundPropTest(recLDesired!FPropertyName_FK, recMarine!SSN_PK) = True Then
                                        Select Case recLDesired!Level
                                            Case 1
                                                intFit = intFit +
                                            Case 2
                                                intFit = intFit +
                                            Case 3
                                                intFit = intFit +
                                            Case 4
                                                intFit = intFit +
                                            Case 5
                                                intFit = intFit +
                                            Case 6
                                                intFit = intFit +
                                        End Select
                                    End If
                                    recLDesired.MoveNext
                                Next l
                                recLDesired.MoveFirst
                            End If
                        End If ' If intFit > 0 Then
                    End If ' If booPEF = True Then
                    recMarClsFit.Edit
                    ' If the marine's PEF is not satisfied by the
school or the marine does not meet
                    ' all the mandatory properties for the school, a
score of zero is assigned.
                    ' Otherwise, a value >= 1 is assigned. A score of
 1 indicates the school had no
                    ' desired properties or the marine did not
satisfy any of the desired properties.
                    ' Later, 1 is subtracted from these scores and
replaced with a small value called
                    ' epsilon. The epsilon is used to prevent a
division by zero problem.
                    recMarClsFit!Fitness = intFit
                    recMarClsFit.Update
                    recMarClsFit.MoveNext
                    ' Prevents counting the entries with zeros since
they are removed later
                    ' Sums up all the fits and gets a count of how
many fits there are
                    If intFit > 0 Then
                        ' The 1 is subtracted from the intFit score
and replaced with a small epsilon
                        sngSumOfFit = sngSumOfFit + CSng(intFit - 1)
#NAME?
                        sngCount = sngCount + 1
                    End If
                    intFit = 0
                    recMarine.MoveNext
                Next j
                ' Prevent dividing by zero
                If sngCount > 0 Then
                    sngAvgFit = sngSumOfFit / sngCount
                    recMarClsFit.MoveFirst
                    ' Update the TEMP_FIT table with the normalized
fitness values
                    For k = 1 To recMarClsFit.RecordCount
                        If recMarClsFit!Fitness > 0 Then
                            recMarClsFit.Edit
                            ' The 1 is subtracted from the stored
fitness score and replaced with a small epsilon
                            sngFitness = CSng(recMarClsFit!Fitness -
1) + sngEpsilon
                            recMarClsFit!Fitness =
CInt(sngMandFitValue + sngFitness * intNormalFactor * sngDesiredWeight /
                            recMarClsFit.Update
                        End If
                        recMarClsFit.MoveNext
                    Next k
                    ' Insert the TEMP_FIT records into the
                    Application.SetOption "Confirm Action Queries",
                    db1.Execute "INSERT INTO MAR_CLS_FIT ( SSN_FK,
MarineIndex, ClassIndex_FK, Fitness ) SELECT TEMP_FIT.SSN_FK,
TEMP_FIT.MarineIndex, TEMP_FIT.ClassIndex_FK, TEMP_FIT.Fitness FROM
TEMP_FIT WHERE TEMP_FIT.Fitness > 0;"
                    Application.SetOption "Confirm Action Queries",
                    strClassNumber = recClass!SCourseNumber_FK
                    strAMOS = recClass!AMOS_FK
                End If
                recMarine.MoveFirst
                If recClass.EOF = False Then
                    recClass.MoveNext
                End If
            End If ' If recMarine.RecordCount > 0 Then
        End If ' If (recClass.SCourseNumber_FK = strClassNumber) And
(recClass.AMOS_FK = strAMOS) Then
    Next i
End If '
```

```

    recMarClsFit.Close

    ' Prepare the AMPL rdm1.dat data file
    Application.SetOption "Confirm Action Queries", False
    db1.Execute "DELETE * FROM TEMP_FIT;"
    ' Specify the number of marines
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK, MarineIndex) VALUES
('param totalMarines :=', " & recMarine.RecordCount & ");"
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
    ' Specify the number of school classes
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK, MarineIndex) VALUES
('param totalClasses :=', " & recClass.RecordCount & ");"
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
    ' Specify the class demand
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES ('param demand
    db1.Execute "INSERT INTO TEMP_FIT ( ClassIndex_FK, Fitness ) SELECT
CLASS.ClassIndex, CLASS.Quota FROM CLASS;"
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
    ' Specify the class penalty
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES ('param penalty
    db1.Execute "INSERT INTO TEMP_FIT ( ClassIndex_FK, Fitness ) SELECT
PENALTY.ClassIndex_FK, PENALTY.Penalty FROM PENALTY;"
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES (';');"
    DoCmd.TransferText acExportFixed, "Rdm1_Data Export Specification",
TEMP_FIT, "C:\RDMn\AmplData\rdm1.dat"

    ' Prepare the AMPL rdm2.dat data file
    db1.Execute "DELETE * FROM TEMP_FIT;"
    db1.Execute "INSERT INTO TEMP_FIT SELECT * FROM MAR_CLS_FIT;"
    db1.Execute "ALTER TABLE TEMP_FIT DROP COLUMN SSN_FK;"
    db1.Execute "ALTER TABLE TEMP_FIT ADD COLUMN SSN_FK text;"
    db1.Execute "INSERT INTO TEMP_FIT (SSN_FK) VALUES ('param fitness
default 0:=');"
    db1.Execute "SELECT * INTO AMPL_TEMP FROM TEMP_FIT ORDER BY SSN_FK
    db1.Execute "INSERT INTO AMPL_TEMP (SSN_FK) VALUES (';');"
    DoCmd.TransferText acExportFixed, "Rdm2_Data Export Specification",
AMPL_TEMP, "C:\RDMn\AmplData\rdm2.dat"
    Application.SetOption "Confirm Action Queries", True

    recMarine.Close
    recClass.Close
    ' Remove the created tables
    db1.Execute "DROP TABLE TEMP_FIT;"
    db1.Execute "DROP TABLE AMPL_TEMP;"
    db1.Execute "Drop Table TEMP_FIT1;"
    db1.Close
    ' Remove the meter from the status bar
    varRetVal = SysCmd(acSysCmdRemoveMeter)

End Sub


Public Sub Quota_Penalty(varUDateBound As Variant)
    Dim varDate
    Dim rec As Recordset, rec1 As Recordset
    Dim strSQL As String, strConvert As String
    Dim i As Integer
    Dim db1 As Database


    ' Create new class quota file
    Application.SetOption "Confirm Action Queries", False

    ' Delete the old penalty and class files
    DoCmd.RunSQL "DELETE * FROM PENALTY"
    DoCmd.RunSQL "DELETE * FROM CLASS"

    ' Query generates new class quota
    DoCmd.SetWarnings (False)
    DoCmd.OpenQuery "qryNewQuota"
    DoCmd.SetWarnings (True)

    ' Delete any values < 0
    DoCmd.RunSQL "DELETE * FROM CLASS WHERE Quota <= 0;"

    Application.SetOption "Confirm Action Queries", True

    ' Create index for the class table and format the FY for RD3 file
    Set db1 = CurrentDb()
    strSQL = "SELECT CLASS.*, SCHOOL.PenaltyFactor FROM SCHOOL INNER JOIN
 CLASS ON (SCHOOL.AMOS_PK = CLASS.AMOS_FK) AND (SCHOOL.SCourseNumber_PK =
 CLASS.SCourseNumber_FK);"
    Set rec = db1.OpenRecordset(strSQL, dbOpenDynaset)
    rec.MoveLast

    If rec.RecordCount > 0 Then
      rec.MoveFirst
      For i = 1 To rec.RecordCount
        rec.Edit
        rec!ClassIndex = i
        ' Converts the fiscal year to a two digit number for RD3 file
        strConvert = CStr(rec!FiscalYear_PK)
        strConvert = Right(strConvert, 2)
        rec!FiscalYear_PK = CInt(strConvert)
        rec.Update
        rec.MoveNext
      Next i
    End If

    ' Create penalty file
    ' First, update the index from the CLASS table
    Application.SetOption "Confirm Action Queries", False
    DoCmd.OpenQuery "qryPenaltyIndex"
    Application.SetOption "Confirm Action Queries", True
    ' Create a record set of the PENALTY table
    strSQL = "SELECT * FROM PENALTY"
    Set rec1 = db1.OpenRecordset(strSQL, dbOpenDynaset)
    If IsNull(varUDateBound) = False Then
      ' Format this date for use in the DateDiff function
      varUDateBound = Format(varUDateBound, "####/##/##")
Module: modFitnessDetermination    Page: 16

      If rec.RecordCount > 0 Then
        rec.MoveFirst
        For i = 1 To rec.RecordCount
          rec1.Edit
          ' Get report date from class record
          varDate = rec!ReportDate
          ' Format the report date for use in the DateDiff function
          varDate = Format(varDate, "####/##/##")
          ' Make the penalty the difference between the two values
          ' This makes the penalty linear, based on date
          rec1!Penalty = DateDiff("d", varDate, varUDateBound) *
rec!PenaltyFactor \ 24
          rec1.Update
          rec.MoveNext
          rec1.MoveNext
        Next i
      End If
    End If

End Sub

Public Sub DeleteTable()
On Error GoTo DeleteTable_Err

    Dim db1 As Database

    Set db1 = CurrentDb()
    ' Delete the table if it already exists
    db1.TableDefs.Delete "PERCENTAGE"
DeleteTable_Exit:
    Exit Sub

DeleteTable_Err:
    Select Case Err.Number
      ' This number occurs if the PERCENTAGE table does not exist.
        Case 3265
          Resume DeleteTable_Exit
        Case Else
          MsgBox Err.Description
          Resume DeleteTable_Exit
          Resume
        End Select

End Sub

Module: modImportExportData
    Code
    Attribute VB_Name = "modImportExportData"
    Option Compare Database
    Option Explicit

    Public Sub ImportBNA()

        DoCmd.TransferText acImportFixed, "BNAExtract", "qryBNAExtract",
C:\ToBeRdm\RdmData\BNA.txt

    End Sub

    Public Sub ImportRD1()

        DoCmd.TransferText acImportFixed, "RD1", "qryMarine",
C:\ToBeRdm\RdmData\RD1.txt

    End Sub

    Public Sub ImportToTarget_MOS() '(ActionQuery As String)

        CurrentDb().Execute "INSERT INTO TARGET_MOS ( CourseNumber_PK,
TargetMOS_PK, MCC ) SELECT DISTINCT BNA_EXTRACT.CourseNumber_PK,
BNA_EXTRACT.TargetMOS_PK, BNA_EXTRACT.MCC FROM BNA_EXTRACT"

    End Sub

    Public Sub Ampl_Result()

        DoCmd.TransferText acImportDelim, "Ampl_Result", "AMPL_RESULT",
C:\ToBeRdm\AmplData\rdm.out, True

    End Sub

    Public Sub AmplData()
        Dim db1 As Database

        Set db1 = CurrentDb()

        ' Prepare the AMPL rdm.dat data file
        Application.SetOption "Confirm Action Queries", False
        db1.Execute "CREATE TABLE CONSTANTS (ConstantName TEXT, intValue
        ' Specify the fit constant
        db1.Execute "INSERT INTO CONSTANTS (ConstantName, IntValue) VALUES
('param fitConstant :=', " & Forms!frmPrepareAndExecuteSolver!cmbFit &
        db1.Execute "INSERT INTO CONSTANTS (ConstantName) VALUES (';');"
        ' Specify the fill constant
        db1.Execute "INSERT INTO CONSTANTS (ConstantName, IntValue) VALUES
('param fillConstant :=', " & Forms!frmPrepareAndExecuteSolver!cmbFill &
        db1.Execute "INSERT INTO CONSTANTS (ConstantName) VALUES (';');"
        DoCmd.TransferText acExportFixed, "Rdm_Data Export Specification",
CONSTANTS, "C:\ToBeRdm\AmplData\rdm.dat"
        Application.SetOption "Confirm Action Queries", True

        ' Remove the created table
        db1.Execute "DROP TABLE CONSTANTS;"
        db1.Close
    End Sub

    Public Sub ExportRD3()

        DoCmd.TransferText acExportFixed, "ExportRD3File", "qryRD3File",
C:\ToBeRDM\RdmData\RD3.txt
        Application.SetOption "Confirm Action Queries", False
        DoCmd.OpenQuery "qryUpdateAssignmentType"
        DoCmd.OpenQuery "qryAppendResultToAssignment"
        Application.SetOption "Confirm Action Queries", True


    End Sub

    Public Sub ArchiveData()

        Dim varTodaysDate
        Dim strSQL As String

        varTodaysDate = Date

        ' Format the date
        varTodaysDate = Format(varTodaysDate, "yyyymmdd")
        ' Convert the date into a number
        varTodaysDate = CLng(varTodaysDate)

        ' Archive the applicable data
        DoCmd.RunSQL "INSERT INTO MARINE_ASSIGN_ARCHIVE SELECT DISTINCTROW
MARINE.SSN_PK, MARINE.LName, MARINE.Initials, MARINE.Gender,
MARINE.DriverLicense, MARINE.SOI, MARINE.TrainingCompany,
MARINE.MCRD_Code, MARINE.PltNumber, MARINE.Citizenship, MARINE.GradDate,
MARINE.SwimQual, MARINE.Obligation, MARINE.Age, MARINE.Education,
MARINE.HighestMath, MARINE.MentalAbility, MARINE.ClearanceEst,
MARINE.Height, MARINE.Vision, MARINE.ColorPerception, MARINE.EthnicGroup,
 MARINE.DLAB, MARINE.EDPT, MARINE.DistrictCode, MARINE.Typing, MARINE.EL,
 MARINE.MM, MARINE.CL, MARINE.GT, MARINE.Component, MARINE.PEF,
MARINE.CivilConviction, MARINE.RSN, MARINE.AFADBD,
ASSIGNMENT.CourseNumber, ASSIGNMENT.AMOS, ASSIGNMENT.MCC,
ASSIGNMENT.ClassNumber, ASSIGNMENT.ReportDate, ASSIGNMENT.FiscalYear,
ASSIGNMENT.AssignmentType FROM MARINE INNER JOIN ASSIGNMENT ON
MARINE.SSN_PK = ASSIGNMENT.SSN_FK WHERE ASSIGNMENT.ReportDate < " &

        ' Delete the applicable data from the Marine and Assignment tables
        DoCmd.RunSQL "DELETE DISTINCTROW MARINE.*, ASSIGNMENT.* FROM MARINE
    DoCmd.RunSQL "DELETE DISTINCTROW MARINE.*, ASSIGNMENT.* FROM MARINE
INNER JOIN ASSIGNMENT ON MARINE.SSN_PK = ASSIGNMENT.SSN_FK WHERE
ASSIGNMENT.ReportDate < " & varTodaysDate & ""

    Select Case MsgBox("Delete unassigned Marines from the database?",
vbYesNo + vbQuestion, "Delete Unassigned Marines")
        Case 6 ' If yes is clicked
            ' Deletes entries in the Marine table without a corresponding
 entry in the Assignment table
            DoCmd.RunSQL "DELETE DISTINCTROW MARINE.*, ASSIGNMENT.SSN_FK
FROM MARINE LEFT JOIN ASSIGNMENT ON MARINE.SSN_PK = ASSIGNMENT.SSN_FK
WHERE (((ASSIGNMENT.SSN_FK) Is Null));"
        Case 7 ' If no is clicked
            ' No action taken
    End Select

End Sub
```

# LIST OF REFERENCES

[1] USMC, Manpower Management Information Systems Branch, Manpower and Reserve Affairs Department. "Statement of Work for the USMC Manpower Model Modernization: Reengineering of the Enlisted Assignment Model and Recruit Distribution Model," September, 1997.

[2] Decision Support Associates Inc. *Users Manual-Recruit Distribution Model.* Developed In-house, Decision Support Associates Inc., 1993.

[3] C.J. Date. *An Introduction to Database Systems.* Reading, MA: Addison-Wesley, 1992.

[4] J. Dumas. *Designing User Interfaces for Software.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[5] CPlex, A Division of ILOG. CPlex Home Page, 1998. [http://www.cplex.com].

[6] R.H. Sprague. "A Framework for the Development of Decision Support Systems." *MIS Quarterly* 4, Pages 1-26, 1980.

[7] Logic Works, Inc. "Logic Works BPWin Tutorial Guide," 1994-1995.

[8] E.F. Codd. "A Relational Model of Data for Large Shared Databanks." *Communications of the ACM* 13, June 1970.

[9] R. Smith, D. Sussman. *Beginning Access 97 VBA Programming.* Wrox Press, 1997.

[10] R. Fourer, D. Gay, B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Boyd & Fraser Publishing Company, 1993.

[11] B. Render, R. Stair. *Quantitative Analysis for Management.* Prentice-Hall Inc., 1997.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center .................................. 2
   8725 John J. Kingman Rd., Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library .................................................. 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Professor Hemant Bhargava (Code SM/BH) ................................ 7
   Systems Management Department
   Naval Postgraduate School
   Monterey, CA 93940-5000

4. Professor Suresh Sridhar (Code SM/SR) ................................ 1
   Systems Management Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor Daniel Dolk (Code SM/Dk) ................................... 1
   Systems Management Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. LT Kevin J. Snoap .................................................... 2
   541 Westway Drive N.W.
   Grand Rapids, MI 49544